Swarm 9

This is the latest documentation, for Swarm 10. To view the documentation for Swarm 9, see the Swarm 9 HTML or Swarm 9 PDF on Caringo Connect.

- Storage SCSP Development
- Content Application Development
- S3 Protocol Interface
- Swarm SDK

Documentation Archive

caringo	© 2005–2019 Caringo, Inc. All rights reserved. Open Source Software Licenses | EULA No part of this material may be reproduced, transmitted, or transcribed without the written consent of Caringo, Inc. Updates may be made to this material and incorporated into later editions.
Current	Swarm 10 Online Documentation | PDF: Documentation (latest) FileFly 3 Online Documentation | PDF: Release Notes, Administration Guide, Community Edition
Prior	Swarm 9 Online Documentation | PDF: Documentation (latest) Swarm 8 Online Documentation | PDF: Release Notes, Guide FileFly 2 Online Documentation | PDF: Release Notes, Guide CloudScaler Online Documentation | PDF: Release Notes, Guide SCSP Proxy | PDF: Release Notes, Overview, OSS Licenses

The online documentation always reflects the latest released version; refer to these PDFs for prior versions:

Swarm 10 Bundle Release Date *	* The PDF might be updated after the bundle is released, but the date in the PDF's filename will not change: it is there to point you to the bundle to which it applies.					
	Swarm Storage	Platform, CSN	ES, Metrics, Search	Gateway, Service Proxy	Content UI	Storage UI
2018 Dec 21 Docs PDF	10.0 New architecture	10.0, 8.3.2 Rolling reboots UI	5.6, 5.0.7, 5.0.7 2.3.3, 2.5, 2.5	5.4, 5.3.0 Metadata translation	6.0 UX improvements	2.0 SSL replication, hardware tools

Swarm 9 Bundle Release Date *	* The PDF might be updated after the bundle is released, but the date in the PDF's filename will not change: it is there to point you to the bundle to which it applies.					
	Swarm Storage	Platform, CSN	ES, Metrics, Search	Gateway, Service Proxy	Content UI	Storage UI
2018 Dec 12 Docs PDF	9.6.4 Intel driver updates: i40e, ixgbe	9.1, 8.3.2	2.3.3, 2.5, 2.5	5.3.3, 5.3.0	5.5.0	1.2.4

2018 Dec 12 Docs PDF	9.6.4 Intel driver updates: i40e, ixgbe	9.1, 8.3.2	2.3.3, 2.5, 2.5	5.3.3, 5.3.0	5.5.0	1.2.4
2018 Aug 31 Docs PDF	9.6.3	9.1, 8.3.2	2.3.3, 2.5, 2.5	5.3.3, 5.3.0 S3 multipart upload/listing fixes	5.5.0	1.2.4
2018 Aug 29 Docs PDF	9.6.3 Support HPE ProLiant (smartpqi)	9.1, 8.3.2	2.3.3, 2.5, 2.5	5.3.2, 5.3.0	5.5.0	1.2.4
2018 Aug 21 Docs PDF	9.6.2 Fixed EC manifest failure to replicate	9.1, 8.3.2 Security improvement	2.3.3, 2.5, 2.5	5.3.2, 5.3.0	5.5.0	1.2.4
2018 Jul 30 Docs PDF	9.6.0	9.1, 8.3 RHEL/CentOS 6.10	2.3.3, 2.5, 2.5	5.3.2, 5.3.0 Bug fixes	5.5.0	1.2.4
2018 Jun 29 Docs PDF	9.6.0 New replication method OSS updates	9.1, 8.3 UEFI support Rolling reboots New CLI	2.3.3, 2.5, 2.5	5.3.1, 5.3.0 New replication method	5.5.0 Object renaming Delete current version	1.2.4 New replication method SwarmNFS 2 support
2018 May 15 Docs PDF	9.5.3 Optimized retires Header filtering OSS updates	9.0, 8.3	2.3.3, 2.5, 2.5 Improved indices	5.2.5, 5.2.3-2	5.4.0 Metadata editor Delete versioned	1.2.3 SwarmNFS export config
2018 Jan 12 Docs PDF	9.4.0 Elasticsearch config script OSS updates (jessie)	9.0, 8.3	2.3.3, 2.4, 2.4-3	5.2.5, 5.2.3-2 Enhanced listing consistency S3 compatibility support	5.3.1 Access policy editor	1.2.1 SwarmNFS 1 support

Swarm Release Notes

Caringo Swarm

- Swarm 10 Highlights
- Swarm Software Bundles

- Swarm Storage Release Notes
- Storage UI Release Notes
- Swarm Platform Release Notes
- CSN Platform Server Release Notes
- Content Gateway Release Notes
- Content UI Release
- SwarmNFS Release Notes
- SDK Release Notes

Swarm 10 Highlights

Swarm combines the scalable software-defined object storage of Swarm Storage with the components to support diverse implementations:

- Platform Server — Node for site-wide management and services
- Storage Cluster — Cluster for Swarm storage nodes
- Elasticsearch — Cluster for search and historical metrics
- Content Gateway — Gateway for cloud-based client access (S3)
- Storage UI — Website for storage cluster management
- Content UI — Website for cloud content management
- SwarmNFS — Optional connector for NFS clients

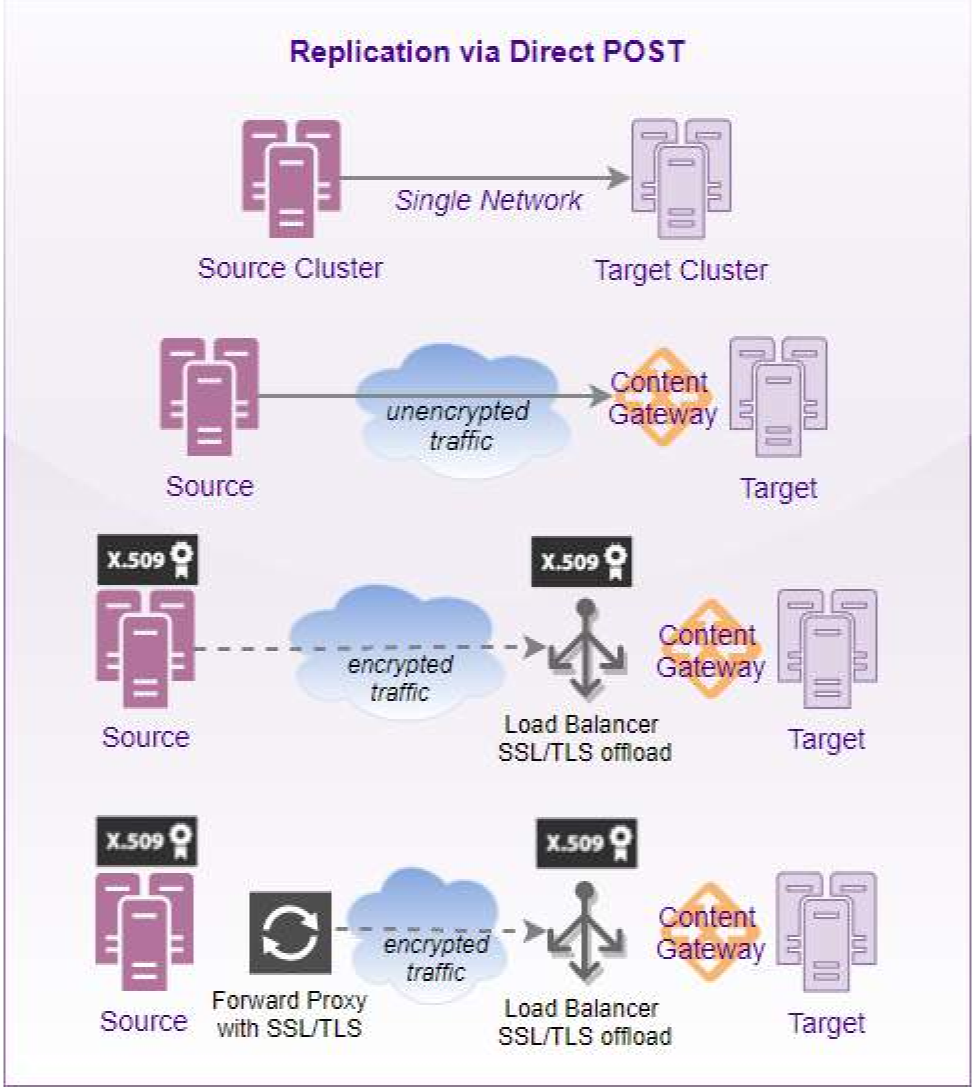
- Swarm 10.0 — launched December 2018

Swarm 10.0 — launched December 2018

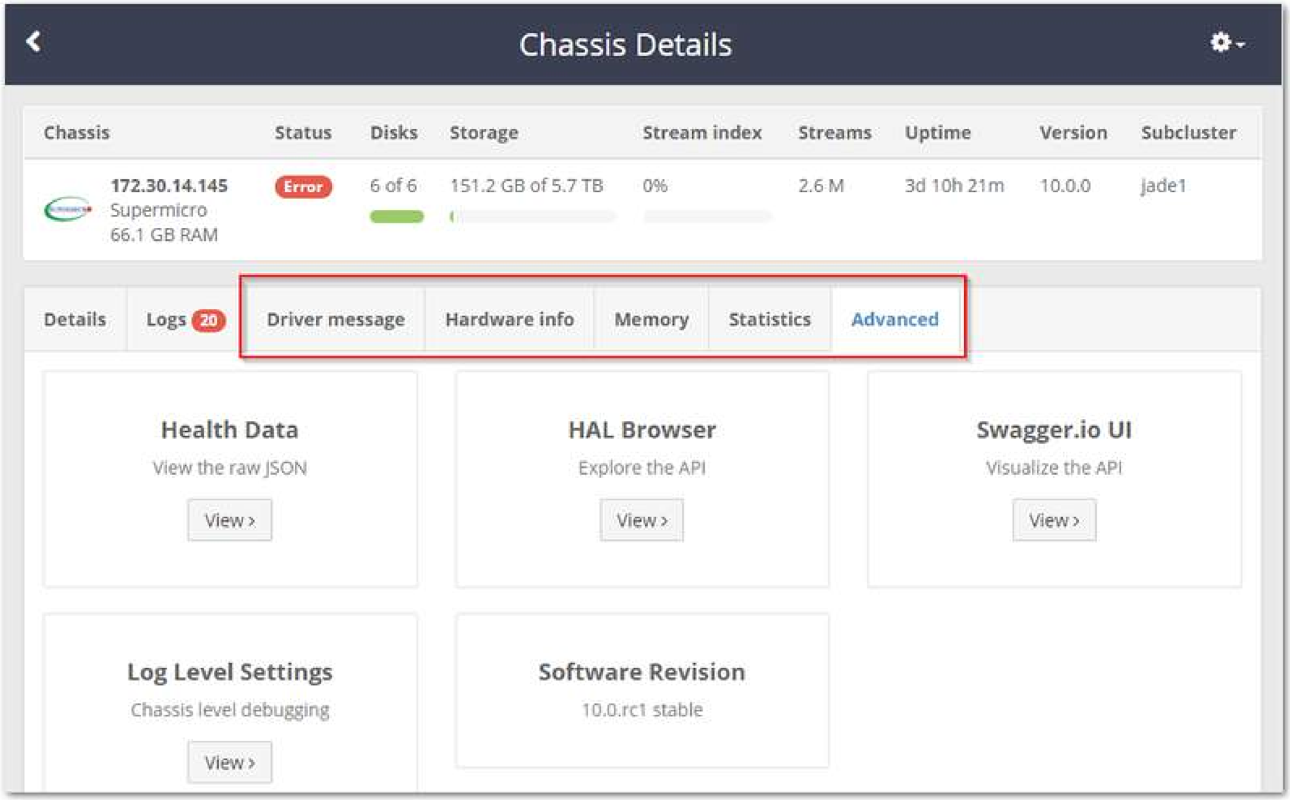
Platform (CSN)	Storage	ES, Search, Metrics	Gateway	Storage UI	Content UI	SwarmNFS
10.0 (8.3)	10.0	5.6.12, 5.0.7, 5.0.7	5.4	2.0	6.0	2.1

- Density-friendly machine addressing — The internal architecture of storage cluster nodes can now easily leverage very dense servers that have a lot of CPU cores and disks. The visible effect of having a single IP per physical or virtual machine is that far fewer IP addresses are needed for deployment, which simplifies network administration, monitoring, and architecture. A new Settings Checker tool identifies configuration changes needed to support upgrading to the new architecture as well as each new version of Swarm, going forward.
- Multiple VLANs and subnets — In order to support larger deployments where storage clusters are spread across multiple racks, rooms, and locations, Swarm now supports multiple VLANs (multiple L3 subnets) within the Platform server and UI. This support lets you more easily grow your cluster from dozens of pieces of hardware to hundreds, and it simplifies your network architecture and administration.
- Remote replication without VPN tunnels — The push-style (direct POST) replication protocol introduced in 9.6 offers better performance and flow control. With 10.0, direct POST now supports SSL/TLS network encryption and standard proxy servers for replication feeds, which eliminates the need for separate VPN tunnels between

clusters. This capability streamlines deployments where encrypted communications are needed over wide-area, untrusted networks, and it streamlines how you can dynamically mirror remote sites:



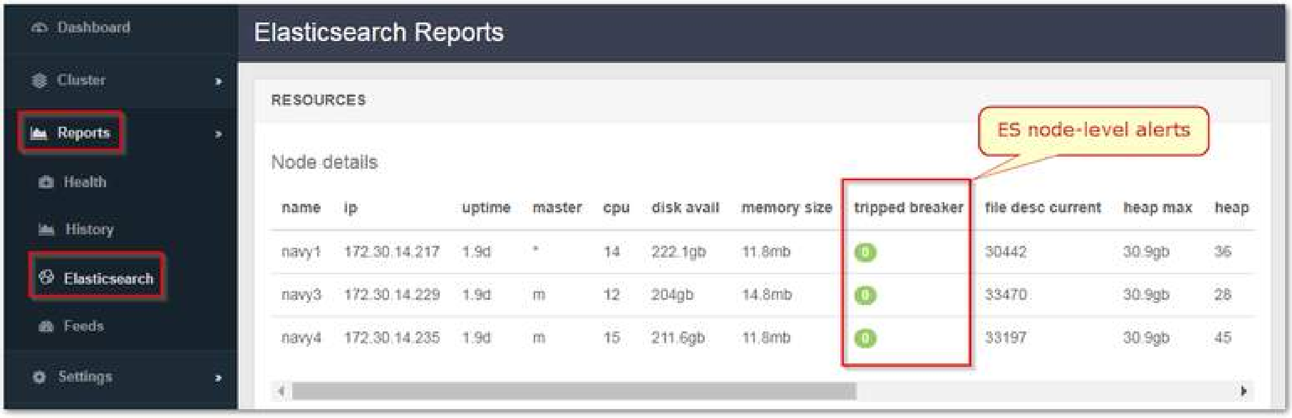
- Elasticsearch major upgrade, atime tracking — Swarm now ships Elasticsearch 5.6, which extends Swarm's built-in metadata searching capabilities and lets you integrate with readily available off-the-shelf tools (such as the ELK stack) that use Elasticsearch for data analytics and monitoring. Once you have reindexed on the new ES schema, you can enable Swarm's new atime (access time) feature, which lets you track content usage and determine candidates for tiering to cold storage.
- Hardware reporting and tools — This new generation of the Swarm UI builds out deep support for hardware management, whether physical or virtual machines. Drilling down to a given machine within Cluster > Hardware, you have new tabs for troubleshooting the chassis: Driver message (Dmesg), Hardware info (Hwinfo), Memory usage, and Statistics detail reports (health processor, Swarm communications, and more). The Advanced tab lets you dynamically change machine-level logging levels and also work with Swarm's management API, both through a hands-on HAL browser and a Swagger visualizer.



Chassis Details

Chassis	Status	Disks	Storage	Stream index	Streams	Uptime	Version	Subcluster
172.30.14.145 Supermicro 66.1 GB RAM	Error	6 of 6	151.2 GB of 5.7 TB	0%	2.6 M	3d 10h 21m	10.0.0	jade1

Elasticsearch clusters also have advanced, detailed troubleshooting help through the Elasticsearch Reports, which span nodes, thread pools, indices, and shards:



Elasticsearch Reports

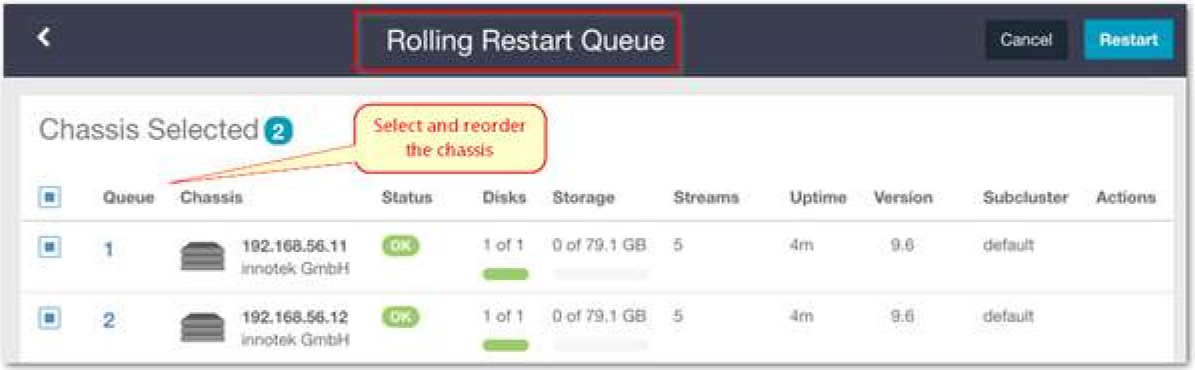
RESOURCES

ES node-level alerts

Node details

name	ip	uptime	master	cpu	disk avail	memory size	tripped breaker	file desc current	heap max	heap
navy1	172.30.14.217	1.9d	*	14	222.1gb	11.8mb	0	30442	30.9gb	36
navy3	172.30.14.229	1.9d	m	12	204gb	14.8mb	0	33470	30.9gb	28
navy4	172.30.14.235	1.9d	m	15	211.6gb	11.8mb	0	33197	30.9gb	45

- Self-service migration from CSN to Zinc — Swarm 10 includes tools and documentation for upgrading from CSN 8.3 to the new generation of Platform server. These enable you to extract the live configuration and run-time information from the CSN Platform 8.3.x server and perform a live migration to the version 9.1 Platform server without downtime for your storage cluster.
- UI enhancements: rolling reboots, UX improvements — Both the Storage and Content UIs have extensive UX improvements for storage cluster administration and content management. A highly requested feature of an orchestrated, unattended rolling reboot of the storage cluster has been included.



Select and reorder the chassis

■	Queue	Chassis	Status	Disks	Storage	Streams	Uptime	Version	Subcluster	Actions
■	1	192.168.56.11 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	4m	9.6	default	
■	2	192.168.56.12 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	4m	9.6	default	

- SwarmNFS update for performance and scaling — The SwarmNFS 2.1 release provides greater single-client performance, security, and overall scaling, supporting parallel ingest of 3 PB+ per month/per instance without special hardware.
- S3 compatibility tracking — Content Gateway continues to keep pace with the evolving S3 protocol changes in order to maintain best-in-class compatibility with applications written for the AWS S3 protocol.
- FileFly 3 — FileFly offloads Windows and NetApp data to all major clouds and writes to multiple destinations simultaneously. 3.0 adds support for the latest Windows and NetApp versions and introduces a free Community Edition (up to 25TB).

Swarm Software Bundles

Swarm distributions bundle together the core components that you need for both implementation and later updates; the latest versions are available on Caringo Connect.

There are two bundles available:

Platform CSN 8.3 Full Install or Update	For CSN environments • CSN 8.3 — installs or updates an existing CSN to 8.3.1 • Swarm Storage RPM, for updating the cluster via the CSN • Elasticsearch — metadata search, usage metering, and operational metrics • Content Gateway — multi-tenant access, delegation, and S3 protocol • User Interfaces — web management of Storage and Content	Flat structure for scripted install/update on a CSN (see CSN Updates and Upgrades). Includes RPMs for a fresh install or upgrade of standalone Content Gateway and Elasticsearch server, as well as both web UIs.

Swarm 10.x Software Bundle	For Platform Server and non-CSN environments • Platform Server — updates an existing 9.x deployment • Storage files for USB flash media booting and building custom PXE boot servers • Elasticsearch — metadata search, usage metering, and operational metrics • Content Gateway — multi-tenant access, delegation, and S3 protocol • User Interfaces — web management of Storage and Content	Contains complete updates of all core components, organized hierarchically by component. See both the top-level README for the manifest of component versions and also the README files specific to each component. Important For new installs of Platform Server, contact Support.

Optional

Optional Swarm client components, such as SwarmNFS and Caringo Drive, have separate distributions.

Getting what you need

1. Log in to Caringo Connect.
2. Download the software. Go to Caringo Swarm (connect.caringo.com/product/caringo-swarm) and download the latest bundle.
    - Whenever a new component version is available (such as a new release of Content Gateway), these bundles are updated and the ZIP file name changes to reflect the release date for the bundle.
3. Download the documentation. From the Documentation panel below the bundle links, select the PDFs link to download the documentation specific to your bundle.
    - The comprehensive PDF that matches all of the components at those versions will be uploaded to the Documentation Archive using a filename that includes that same release date for its matching bundle.

    Documentation

    Caringo Connect's online HTML documentation is continually updated to the current release; because the PDF is a snapshot that is tied to a bundle release, consider it the definitive source for that bundle.

    Caringo Support's searchable Knowledge Base contains both technical articles and the very latest documentation, which might be newer than your installed version.

4. Expand your bundle. In the top-level directory of the bundle, locate and read the `README.txt` for the version guidance on using the bundle.
5. Open your bundle PDF and see the Release Notes for each component, which include upgrade instructions as well as changes and watch items.

Swarm Storage Release Notes

> Important
> If you are upgrading from a prior version, review the new features and upgrade impacts for each version since the version from which you are upgrading.

Swarm Storage 10.0 Release

New Features

- Density-friendly machine addressing — With Storage 10.0, Swarm has a new internal architecture so that it can easily leverage very dense servers that have a lot of CPU cores and disks. The visible effect of having a single IP per physical or virtual machine is that far fewer IP addresses are needed for deployment, which simplifies network administration, monitoring, and architecture. With Swarm 10's new single-IP address architecture, every physical or virtual machine only requires one IP address, and each Swarm "node" now simply refers to the machine that hosts it, because only a single instance of Swarm software runs on it.
- Settings Checker — A new Storage Settings Checker tool, bundled with the Caringo Support Tools, identifies configuration changes needed to support the new Swarm 10 architecture. For this and all future upgrades of Swarm, running the settings checker before upgrading or during troubleshooting will greatly improve Support's ability to keep your configuration pruned and tuned. See Storage Settings Checker.
- Remote replication without VPN tunnels — The push-style (direct POST) replication protocol introduced in 9.6 offers better performance and flow control. With 10.0, direct POST now supports SSL/TLS network encryption and standard proxy servers for replication feeds, which eliminates the need for separate VPN tunnels between clusters. This capability streamlines deployments where encrypted communications are needed over wide-area, untrusted networks. See Replicating Feeds over Untrusted Networks.
    - Swarm now supports using SSL for remote replication data transfer. This configuration requires an SSL offload proxy (such as HAProxy) in the target cluster environment. (SWAR-7826)
    - For sites using SSL with remote replication, Swarm allows the establishment of trusted certificates (public keys) that may be self-signed. (SWAR-8080) See Adding a Trusted Certificate to Swarm.
    - Swarm now lets you put a forward proxy to the source cluster into the replication path. (SWAR-8025)
- Elasticsearch 5.6 — Swarm now ships with Elasticsearch 5.6, which extends Swarm's built-in metadata searching capabilities and lets you integrate with readily available off-the-shelf tools (such as the ELK stack) that use Elasticsearch for data analytics and monitoring. Swarm remains backwards compatible with Elasticsearch 2.3.3 to allow for a well-timed migration to the new version, which requires reindexing of your search data. See Migrating from Elasticsearch 2.3.3.
- Time of access (atime) tracking — Once you have begun reindexing your search data on the new Elasticsearch 5.6 schema, you can enable Swarm's new atime (access time) feature, which gives your client applications a

way to track content usage and determine candidates for deletion or tiering to cold storage. See Time of Last Access - atime.

- Improved administration
    - Security: The SNMP password is separated from the security.administrators setting into its own setting (`snmp.rwCommunity`), as part of better administrative password handling. (SWAR-8097)
        - See Swarm Passwords.
    - Search:
        - The new "`versioned`" search query argument lets you filter objects based on their versioning status. It checks the value of the `Castor-System-IsVersioned` header, which captures whether versioning was enabled (true) or suspended (false) in the object's context at the time it was written. (SWAR-6851)
        - When indexing quoted strings, Swarm now preserves any existing RFC-2047 encoding. (SWAR-8089)
    - Logging:
        - To help troubleshoot a failure to boot, the Swarm console can now display the startup log ("castor_init"). Under Diagnostics, select "11. Castor Startup log". (SWAR-8070)
        - Upon boot, dmesg is now logged and preserved automatically as an historical record of the starting state of the machine. (SWAR-8040)
- OSS Updates — Storage 10.0 includes significant updates to third-party components. See Third-Party Components for 10.0 for the complete listing of packages and versions.
    - Upgraded kernel version to 4.14.53, linux firmware to 1.174, and bnx2 firmware to firmware-bnx2_20161130-3-bpo8+1. (SWAR-8048)
    - Added kernel support for Virtio devices in order to better support KVM virtual machines. (SWAR-8043)
    - Added the Broadcom NetXtreme driver, to support newer adapters. (SWAR-7627)
    - Updated third-party packages Twisted 18.4.0 and Boost 1.67. (SWAR-7981)

Additional Changes

These items are other changes and improvements including those that come from testing and user feedback.

- SCSP fixes
    - When a write fails because volumes are not available, Swarm returns a 507 Insufficient Storage error rather than 4xx client errors. (SWAR-8073)
    - A multipart APPEND on a previously renamed object dropped the generation NID header, which prevented any segments written before the rename from finding the manifest, causing them to be deleted as orphan s. (SWAR-8300)
    - Swarm no longer allows an object that has a multipart APPEND or PATCH initiated but then is updated by a POST or PUT to complete its multipart write, because those segments would be unable to find the new manifest, causing them to be deleted as orphans. (SWAR-8192)
- Feed fixes
    - During replication through Gateway, when Gateway disallowed replication on particular objects for whatever reason, Swarm treated those events as failures to be retried instead of allowing the feed to block. (SWAR-8060)
    - On the Feeds definition page of the legacy Admin Console (port 90), deselecting the "Propagate Deletes" checkbox had no effect on propagation. (SWAR-8076)
- Boot fixes

- Rebooting a recovered volume could erroneously trigger a recovery (FVR). Upgrading to Swarm 10 and completing a full HP cycle will prevent this from occurring. (SWAR-8081)
- Errors (Failed to start Create Volatile Files and Directories) appeared on the Swarm console during startup. (SWAR-7762)
- When using DHCP IP assignment (no assigned `network.ipAddress`), Swarm 9.6+ failed to boot. (SWAR-8212)

Upgrade Impacts

These items are changes to the product function that may require operational or development changes for integrated applications.

- Upgrading Elasticsearch — You may continue to use Elasticsearch 2.3.3 with Storage 10.0 until you are able to move to 5.6 (see Migrating from Elasticsearch 2.3.3). Support for ES 2.3.3 will end in a future release.
- Configuration Settings — Be sure to run the Storage Settings Checker to identify these and other configuration issues.
  - Changes for the new single-IP dense architecture:
    - `network.ipAddress` — multiple IP addresses now disallowed
    - `chassis.processes` — removed; multi-server configurations are no longer supported
    - `ec.protectionLevel` — new value "volume"
    - `ec.subclusterLossTolerance` — removed
  - Changes for security (see next section)
    - `security.administrators, security.operators` — removed 'snmp' user
    - `snmp.rwCommunity, snmp.roCommunity` — new settings for 'snmp' user
    - `startup.certificates` — new setting to hold any and all public keys
  - New settings:
    - `disk.atimeEnabled`
    - `health.parallelWriteTimeout`
    - `search.pathDelimiter`
- Required SNMP security change — Remove the `snmp` key from the `security.administrators` setting, and update `snmp.rwCommunity` with its value. Nodes that contain only the `snmp` key in the `security.administrators` setting will not boot. If you changed the default value of the snmp key in the `security.operators` setting, update `snmp.roCommunity` with that value and then remove the `snmp` key from `security.operators`. In the `security.operators` setting, '`snmp`' is a reserved key, and it cannot be an authorized console operator name. (SWAR-8097)
- EC protection
  - Best practice: Use `ec.protectionLevel=node`, which distributes segments across the cluster's physical/virtual machines. Do not use `ec.protectionLevel=subcluster` unless you already have subclusters defined and are sure that you have enough to support your specified EC encoding. A new level, `ec.protectionLevel=volume`, allows EC writes to succeed if you have a small cluster with fewer than (k+p)/p nodes. (Swarm always seeks the highest protection possible for EC segments, regardless of the level you set.)
  - Optimize your hardware for EC by ensuring that there are more than k+p subclusters/nodes (as set by `ec.protectionLevel`); for example, with `policy.ecEncoding=5:2`, you need at least 8 subclusters/nodes. When Swarm cannot distribute EC segments adequately for protection, EC writes can

fail despite ample free space. (SWAR-7985)

  - Setting `ec.protectionLevel=subcluster` without creating subclusters (defining `node.subcluster` across sets of nodes) causes a critical error and lowers the protection level to 'node'. (SWAR-8175)
- Small clusters — If you have 10 or fewer Swarm nodes (never use fewer than 3 in production), verify the following settings.
  Important: If you need to change any, do so before upgrading to Swarm 10.
  - policy.replicas — The `min` and `default` values for numbers of replicas to keep in your cluster must not exceed your number of nodes. For example, a 3-node cluster may have only `min=2` or `min=3`.
  - EC encoding and protection — For EC encoding, verify that you have enough nodes to support your cluster's encoding (`policy.ecEncoding`). For EC writes to succeed with fewer than (k+p)/p nodes, use the new level, `ec.protectionLevel=volume`.
  - Best practice: Keep at least one physical machine in your cluster beyond the minimum number needed. This allows for one machine to be down for maintenance without compromising the constraint.
- "Cluster in a box" — Swarm supports a "cluster in a box" configuration as long as that box is running a virtual machine host and Swarm instances are running in 3 or more VMs. Each VM boots separately and has its own IP address. Follow the recommendations for small clusters, substituting VMs for nodes. If you have two physical machines, use the "cluster in a box" configuration, but move to direct booting of Swarm with 3 or more.
- Offline node status — Because Swarm 10's new architecture reduces the number of IP addresses in your storage cluster, you might see the old IPs and subclusters reporting as Offline nodes until they timeout in 4 days (`crier.forgetOfflineInterval`), which is expected.

Watch Items and Known Issues

The following operational limitations and watch items exist in this release.

- When you try to view Health Data (the raw JSON of the health report) on the Advanced tab of the Chassis Details page, the node may become temporarily unresponsive. (SWAR-8349)
- While a reboot of a storage node is in progress, it may be reported to be in an unknown state rather than in maintenance mode. (SWAR-8348)
- If you wipe your Elasticsearch cluster, the Storage UI will show no NFS config. Contact Support for help repopulating your SwarmNFS config information. (SWAR-8007)
- If you delete a bucket, any incomplete multipart upload into that bucket will leave its parts (unnamed streams) in the domain. To find and delete them, use the s3cmd utility (search the Support site for "s3cmd" for guidance). (SWAR-7690)
- Dell DX hardware will have less chassis-level monitoring information available via SNMP. If this is a concern, contact Support. (SWAR-7606)
- Logs showed the error "FEEDS WARNING: calcFeedInfo() couldn't find realm for xxx". The root cause is fixed; if you received such warnings, contact Support so that your issue can be resolved. (SWAR-7556)
- With multipath-enabled hardware, the Swarm console Disk Volume Menu may erroneously show too many disks, having multiplied the actual disks in use by the number of possible paths to them. (SWAR-7248)

Upgrading from 9.x

> Important
>
> Do not begin the upgrade until you complete the following:

1. Plan upgrade impacts — Review and plan for the 10.0 upgrade impacts (above) and the impacts for each of the releases since the version you are running. That is, if you are upgrading from 9.2, review 9.3, 9.4, through to the current release.
2. Finish volume retires — Do not start any elective volume retirements during the upgrade. Wait until the upgrade is complete before initiating any retires.
3. Run checker script — Swarm 10 includes a migration checker script to run before upgrading from Swarm 9; it reports configuration setting issues and deprecations that must be addressed. (SWAR-8230) See Storage Settings Checker.

If you need to upgrade from Swarm 8.x or earlier, contact Support for guidance.

1. Download the correct bundle for your site. Swarm distributions bundle together the core components that you need for both implementation and later updates; the latest versions are available on Caringo Connect.
   There are two bundles available:
   - Platform CSN 8.3 Full Install or Update (for CSN environments) — Flat structure for scripted install/update on a CSN (see CSN Updates and Upgrades).
   - Swarm 10 Software Bundle (Platform 9.x and custom environments) — Contains complete updates of all core components, organized hierarchically by component.

     Note

     For new installs of Platform Server, contact Support.

     Optional Swarm client components, such as SwarmNFS and Caringo Drive, have separate distributions.
2. Download the comprehensive PDF of Swarm Documentation that matches your bundle distribution date, or use the online HTML version:
   https://connect.caringo.com/content/documentation-and-training
3. Choose your type of upgrade. Swarm supports rolling upgrades (a single cluster running mixed versions during the upgrade process) and requires no data conversion unless specifically noted for a particular release. This means that you can upgrade without scheduling an outage or bringing down the cluster. Just restart your nodes one at a time with the new version and the cluster will continue serving applications during the upgrade process.
   - Rolling upgrade: Reboot one node at a time and wait for its status to show as "OK" in the UI before rebooting the next node.
   - Alternative: Reboot the entire cluster at once after the software on all USB flash drives or the centralized configuration location has been updated.
4. Choose whether to upgrade Elasticsearch 2.3.3 at this time.
   - To upgrade to Elasticsearch 5.6 with an existing cluster, you must reindex your Search data and migrate any Metrics data that you want to keep. See Migrating from Elasticsearch 2.3.3 for details. (SWAR-7395)
5. Note these installation issues:
   - The elasticsearch-curator package may show an error during an upgrade, which is a known curator issue. Workaround: Reinstall the curator: `yum reinstall elasticsearch-curator` (SWAR-7439)
   - Do not install the Swarm Search RPM before installing Java. If Gateway startup fails with "Caringo script plugin is missing from indexer nodes", uninstall and reinstall the Swarm Search RPM. (SWAR-7688)
   - During a rolling upgrade from 9.0.x–9.2.x, you may see intermittent "WriterMissingRemoteMD5 error token" errors from a client write operation through the Gateway or on writes with gencontentmd5 (or the

equivalent). To prevent this, set `autoRepOnWrite=0` during the upgrade and restore `autoRepOnWrite=1` after it completes. (SWAR-7756)

6. Review the Application and Configuration Guidance.

Third-Party Components for 10.0

```
Linux Distribution: jessie
Linux Kernel: 4.14.53
iperf3 version: 3.1.3-1
libiperf0 version: 3.1.3-1
megacli version: 8.07.14-2
openvswitch-common version: 2.3.2-1
openvswitch-switch version: 2.3.2-1
Linux Firmware: 1.174
Zope Interface version: 4.5.0
Incremental version: 17.5.0
pyOpenSSL version: 18.0.0
service_identity version: 17.0.0
Twisted version: 18.4.0
Egenix MX Base version: 3.2.9
Simplejson version: 3.1.3
TXES version: 0.1.4
Newt version: 0.52.20
Ipaddress version: 1.0.22
NTP version: 4.2.8p10
Irqbalance version: 1.1.0
Argparse version: 1.2.1
Zbase32 version: 1.1.5
Pyutil version: 3.1.0
Zfec version: 1.4.22
Guppy version: 0.1.10.r95
Boost version: 1_67_0
Werkzeug version: 0.9.6
Klein version: 15.0.0
CAStor SDK version: 6.1.5.1-py2.5
Requests version: 90f3842
Python Dateutil version: 1.5
SNMP version: 5.7.3
Yajl version: 2.1.0-0-ga0ecdde
Intel ixgbe driver version: 5.3.7
Intel igb driver version: 5.4.0-k
Intel e1000e driver version: 3.2.6-k
Intel ixgbevf driver version: 4.1.0-k
Intel i40e driver version: 2.4.10
Intel i40evf driver version: 3.0.0-k
Cryptogrpahy version: 2.2.2
Pip list:
asn1crypto (0.24.0)
attrs (18.2.0)
Automat (0.7.0)
castorsdk (6.1.5.1)
certifi (2018.10.15)
cffi (1.11.5)
chardet (2.3.0)
colorama (0.3.2)
constantly (15.1.0)
cryptography (2.3.1)
defusedxml (0.4.1)
docutils (0.12)
```

```
egenix-mx-base (3.2.9)
enum34 (1.1.6)
functools32 (3.2.3.post2)
guppy (0.1.10)
html5lib (0.999)
hyperlink (18.0.0)
idna (2.7)
incremental (17.5.0)
ipaddress (1.0.22)
jsonschema (2.5.1.post7)
klein (15.0.0)
mock (1.0.1)
ndg-httpsclient (0.3.2)
Pillow (2.6.1)
pip (9.0.1)
ply (3.4)
pyasn1 (0.4.4)
pyasn1-modules (0.2.2)
pycparser (2.19)
Pygments (2.0.1)
pyOpenSSL (18.0.0)
python-apt (0.9.3.12)
python-dateutil (1.5)
python-debian (0.1.27)
python-debianbts (1.11)
pyutil (3.1.0)
reportbug (6.6.3)
requests (2.13.0)
roman (2.0.0)
service-identity (17.0.0)
setuptools (18.4)
simplejson (3.1.3)
six (1.11.0)
SOAPpy (0.12.22)
treq (0.2.0)
Twisted (12.2.0)
txes (0.1.4.dev0)
urllib3 (1.9.1)
Werkzeug (0.9.6)
wheel (0.24.0)
wstools (0.4.3)
zbase32 (1.1.5)
zfec (1.4.22)
zope.interface (4.0.1)
Python version: 2.7.9
megaraid_sas driver version: 07.702.06.00-rc1
mpt2sas driver version: 15.100.00.00
mpt3sas driver version: 15.100.00.00
bnx2 driver version: 2.2.6
bnx2x driver version: 1.712.30-0
curl version:
nload version:
hpsa driver version: 3.4.20-0
libpng version: 1.2.8
LILO version: 22.7.1
Yajl-py version: 2.1.1
```

```
Mock library version: 1.0.1
treq version: 0.2.0
pstat.py version: 0.4
```

Application and Configuration Guidance

Follow this guidance when developing Swarm applications or configuring Swarm clusters.

- Changing Drive Controllers. Administrators must not move Swarm storage drives between drive array controller types after the drive has been formatted by Swarm. Each controller reports available drive space to Swarm that is matched with the controller. For example, many controllers claim the last section of the drive, reducing the total available drive space. If you switch your drives with another controller, the new controller may claim additional drive space that is not reported to Swarm. As a result, Swarm may attempt to write data to non-existing drive space, generating I/O errors.
- Indexer Query Arguments. The Indexer searching syntax allows for repeated constraints on a field name in the HTTP query string. If you are having problems using this, check that your HTTP client library is passing all instances of the repeated name and not consolidating the repeats into one name/value pair.
- SNMP behavior with snmpwalk and snmpgetnext. To be consistent with standard SNMP behavior, the following changes were made to Swarm's SNMP agent:
  - All scalar object IDs (OIDs) end with .0
  - All table OIDs x for row r will be returned as x.r from a snmpwalk or snmpgetnext. As a result, you might need to change any custom applications that use snmpwalk or snmpgetnext.
  - Note the following about Swarm Management Information Base (MIB) return values in this release:
    - The values for system.sysName and system.systemLocation now match values in the node or cluster configuration file instead of defaults that were returned in earlier Swarm versions. The values are read-only. (The Swarm version 6.0 configuration variable names are snmp.sysName an d snmp.sysLocation, respectively; in earlier Swarm versions, these variables were named snmpSys Name and snmpSysLocation.)
    - The value for system.sysDescr returns a value like Linux Caringo 2.6.38.8 instead of the Swarm version and revision. Linux Caringo 2.6.38.8 is a standard SNMP return value and will be used from now on.
- Known issues with Windows 200x Server time synchronization. Caringo strongly recommends you configure your cluster to use Network Time Protocol (NTP) as documented about the timeSource. You cannot use Windows 200x servers as your NTP time source. As discussed in Microsoft KB article 939322, Windows servers are not reliable enough to provide highly accurate time synchronization. As an alternative to using time synchronization available in Windows servers, consider the following possibilities:
  - Use NTP servers available on the internet, such as the servers discussed on the NTP Pool Project page. You might need to open a port in your firewall to enable the cluster to use external NTP servers.
  - Use an open source NTP package such as the Windows based NTP Time Server Monitor.
  - Use a commercial Windows NTP package or deploy a dedicated NTP hardware solution in your network.
- Virtual Deployments Administrators wanting to run Swarm in a virtual environment such as VMware must contact their Support representative for restrictions and guidelines prior to deploying Swarm in a VM.
- Duplicate domain and bucket creation in mirrored clusters. In a certain cluster configuration referred to as active-active, do not attempt to create the same domain or same bucket in the same domain in each cluster. Instead, create the domain or bucket on one cluster and wait for it to be replicated to the other cluster. Failure to do so results in the domain or bucket with the latest creation date taking precedence and objects contained in the other domain or bucket being inaccessible.

- Use curl 7.20.1 or later. If you use curl with Swarm, and you use the authorization feature, you must use curl 7.20.1 or later. There are known issues with earlier curl versions.
- Consumer-Grade Drives. Some non-enterprise-class drives have lengthy error recovery logic. When an error occurs on these types of drives, it might take minutes for a read or write operation to complete. In these cases, the client could see very long response times, or it might even see a socket timeout if the delay is too long. Many enterprise or server grade drives are designed to return errors within a limited period, allowing recovery or rebuild operations to begin immediately and to eliminate the lengthy delays on I/O operations.

> Important
>
> Caringo does not support consumer-grade drives in high-demand environments.

- Avoiding Client Timeouts with Large Objects. Client operations with large objects (1 GB or larger) can take several seconds or more, depending on object size. Clients that support large object operations should set their socket timeouts accordingly to avoid client timeout errors.
- Time Clock Synchronization for Client Servers. When formatting storage policies in lifepoint headers, it is very important that the local clock on the machine creating the lifepoint be reasonably accurate so the end dates of the lifepoints reflect the true UTC time. The Swarm cluster itself can (and should) synchronize itself to an accurate time source. If the client mistakenly specifies the wrong end date in an object's storage policy, perhaps because its local clock is set incorrectly, there could be unintended consequences, including premature deletion, when Swarm enforces that policy.
- Replica Terminology. The term replica has special meaning in the context of Swarm. All instances of an object stored in a Swarm cluster are identical – there is no original. Therefore, saying there are two replicas of an object means there are exactly two identical instances (not an original plus two copies).
- Not Found Errors in a Busy Cluster. A READ, INFO, UPDATE, or DELETE request to a heavily loaded cluster might rarely result in a 404 Not Found response, even if the requested object is present in the cluster. If your client has a priori knowledge that a certain object is stored, it should retry the request until it succeeds. Normally, only a single retry will be required.
- Network Interface Required. Every Swarm node requires a working network connection. If a network cable is unplugged or if the network is not operational at any time during or after startup, neither SNMP nor SCSP are available. Therefore, the only indication of this condition is in the attached console (if there is one), where errors such as "Network Unreachable" display. Once the connection is restored, Swarm recovers and continues running. SNMP or ping monitoring should be implemented to verify proper network connectivity among Swarm nodes.
- HTTP Client Library Limitations. Some HTTP client libraries, including Microsoft .NET HttpWebRequest and httplib in Python, do not handle the Expect: 100-continue header properly. A client should include this header when writing content larger than 64 KB and wait after sending the initial headers for a response before sending additional content to Swarm. Possible responses at that point are a redirect (301 or 307), an error response, or the 100 Continue response, which means continue sending data now. Per the HTTP specification, it is not permissible to continue sending data before receiving a 100-continue response from the server when an Expect: 100-continue header has been included in the request. These issues have been resolved in the Swarm Software Development Kit but integrators writing a non-SDK client should be aware of these limitations.
- Available Drive Space. The available drive space reported by the Swarm Management Console and SNMP is an accurate estimate of the amount of usable space available on a volume or node. However, the calculation of this value takes into account a number of internal considerations that might not be immediately visible to an administrator. For example, Swarm reserves space on a volume equal to two times the size of the largest object or EC segment stored on a given volume to allow for continuous defragmentation. This means the very first object or EC segment stored on a volume appears to consume more space than you might otherwise expect it to

consume. The UUID of the object or EC segment used to reserve defrag space is available in the CARINGO-CASTOR-MIB.

- Available Index Slots. The management console may slightly over-estimate the number of available index slots in a node. For capacity planning purposes, use the estimates provided in the memory table.
- Retire in Small Clusters. To retire a node or volume, there must be at least two suitable nodes in the cluster that have storage space available. Volume-less nodes and nodes that are themselves retired or retiring do not count as suitable nodes. In a multi-server configuration, suitable nodes can include other nodes running in the same physical chassis as the retiring node or volume.

Storage UI Release Notes

If you are upgrading from a prior version, review the release notes for each version since the version from which you are upgrading.

> For upgrade steps, see Installing the Storage UI.

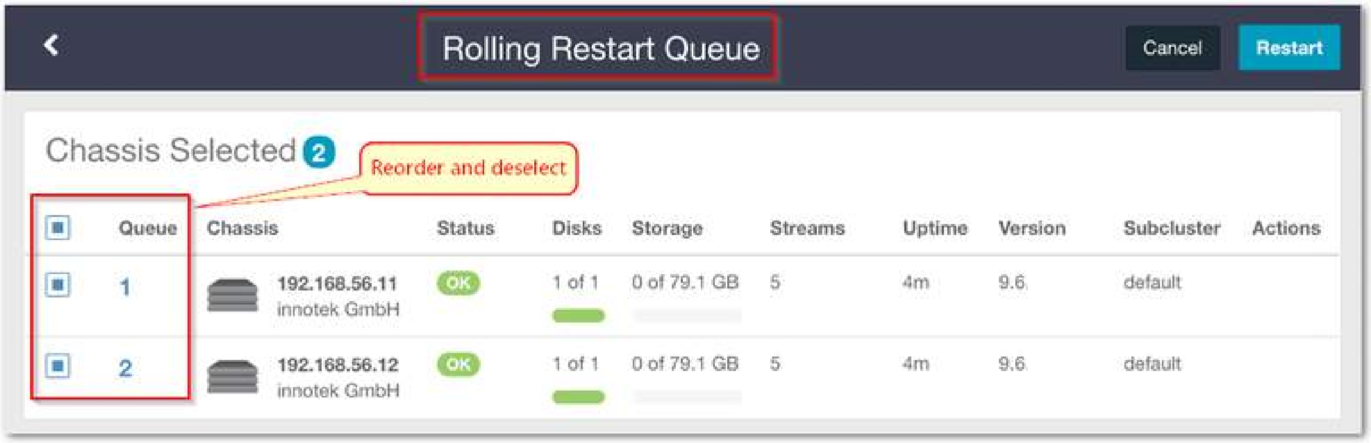
Changes in Storage UI 2.0

> Required integrations
> This version requires Swarm 10.0 or later, Gateway version 5.4 or later, and SwarmNFS version 2.1 or later, if used.

Hardware Management

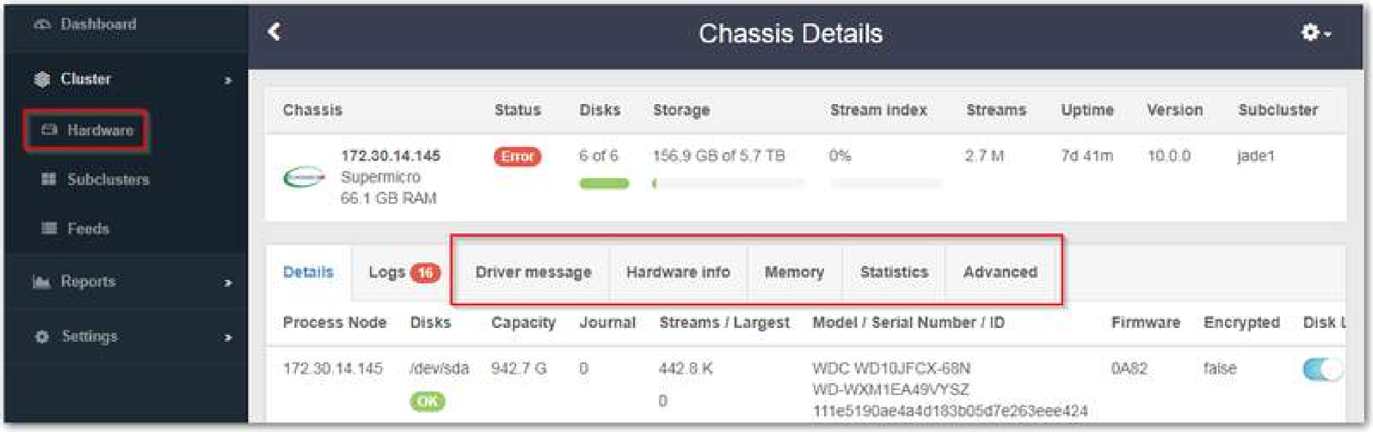
Aligning with the new architecture of Swarm Storage 10, Storage UI 2.0 has been extensively expanded to support the monitoring and administration of Swarm implementations, replicating the rich functionality of the legacy Admin Console (which is deprecated) and adding visibility into Swarm's management API.

Rolling Restarts — With Platform Server installed, you can choose to perform a Rolling Restart, so that the cluster remains fully operational, with chassis going offline one at a time to avoid service interruption. The Rolling Restart Queue lets you reorder and remove chassis from the queue, monitor the progress, and cancel queued restarts. (UIS-588)



■	Queue	Chassis	Status	Disks	Storage	Streams	Uptime	Version	Subcluster	Actions
■	1	192.168.56.11 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	4m	9.6	default	
■	2	192.168.56.12 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	4m	9.6	default	

Chassis Details page features:

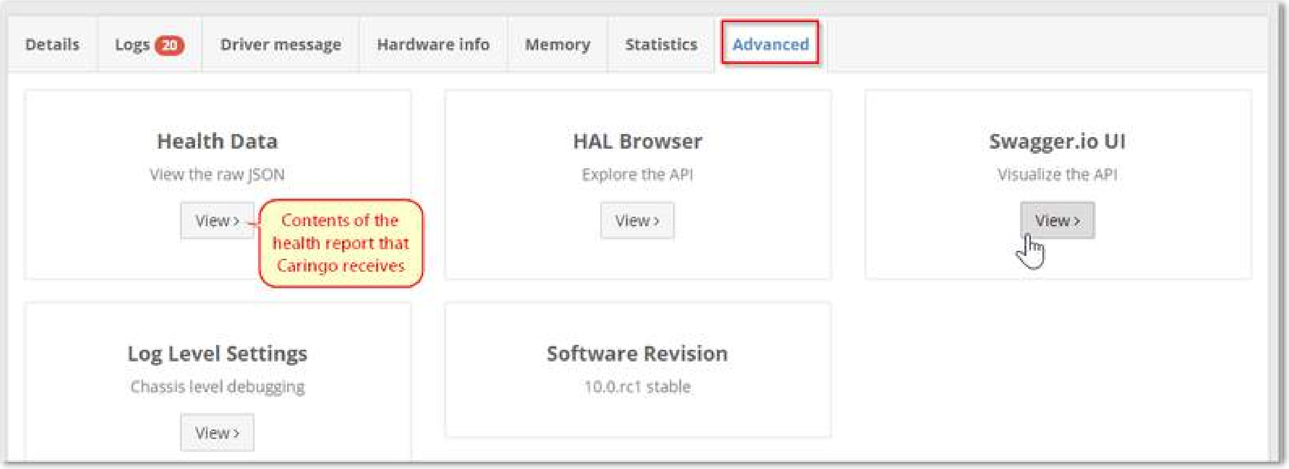


Chassis Details

Chassis	Status	Disks	Storage	Stream index	Streams	Uptime	Version	Subcluster
172.30.14.145 Supermicro 66.1 GB RAM	Error	6 of 6	156.9 GB of 5.7 TB	0%	2.7 M	7d 41m	10.0.0	jade1

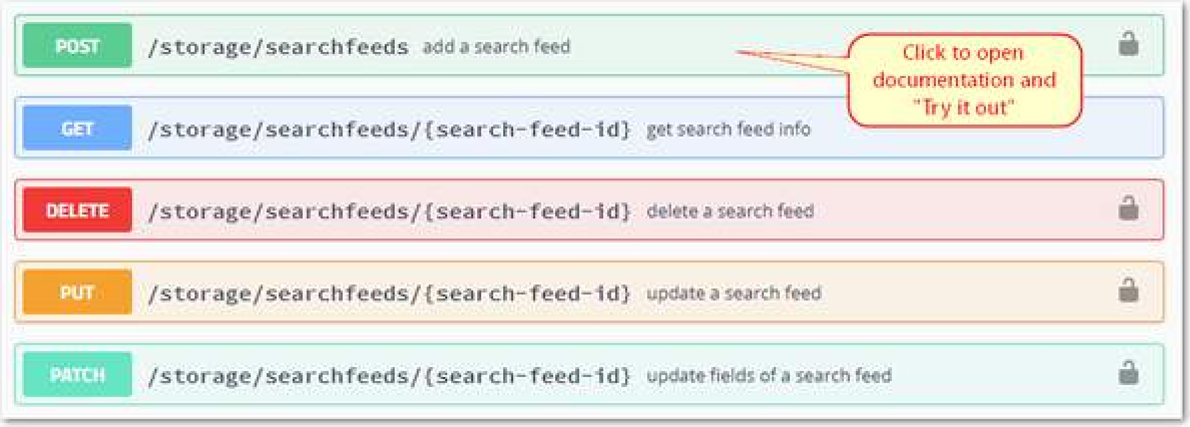
Process Node	Disks	Capacity	Journal	Streams / Largest	Model / Serial Number / ID	Firmware	Encrypted	Disk L
172.30.14.145	/dev/sda OK	942.7 G	0	442.8 K 0	WDC WD10JFCX-68N WD-WXM1EA49VYSZ 111e5190ae4a4d183b05d7e263eee424	0A82	false	

- Details Tab
  - The main Details tab includes counts of each disk's streams and the size of its largest stream. Watching these counts helps you monitor the progress of disks being retired. (UIS-533)
  - You can now retire disks individually as well as retire the machine (chassis) as a whole. Disk-level retires are useful for targeting bad (slow) disks and for working around having too limited capacity for retires of entire chassis. (UIS-544)
  - When one or more disks is in the process of retiring, the affected chassis now shows that status, for improved visibility and tracking. (UIS-749)
- Logs Tab
  - The Logs tab now lets you clear out the logs when they are no longer needed. (UIS-616)
- Driver Message Tab (new)
  - Driver Message displays the output from the dmesg command, which prints the message buffer of the kernel. (UIS-799)
- Hardware Info Tab (new)
  - Hardware Info displays the output of the hwinfo hardware detection tool. (UIS-799)
- Memory Tab (new)
  - The Memory tab reports details of memory usage on the specific machine, to help with capacity planning and analysis. (UIS-723)
- Statistics Tab (new)
  - The Statistics tab rolls up a detailed, expandable report that combines Health Processor (HP), Communications (cluster network), and Memory usage counts and values, to help with analysis and troubleshooting. (UIS-886)
- Advanced Tab (new)



Contents of the health report that Caringo receives

  - You can view the cluster's Health Data, which is the raw JSON content of the health report that your cluster sends to Caringo Support. (UIS-838)
  - You can view and change the Log Level Settings dynamically, to simplify machine-level debugging. (UIS-835)
  - You can verify the Software Revision of Swarm Storage that is running on the specific machine, which is useful for managing rolling upgrades. (UIS-802)
  - You can use an embedded HAL Browser so that you can explore the complete API for Swarm storage management dynamically. (UIS-836)
  - You can access the Swagger.io UI. This API visualization tool also lets you explore Swarm's API for managing the storage cluster. (UIS-837)



POST /storage/searchfeeds add a search feed

GET /storage/searchfeeds/{search-feed-id} get search feed info

DELETE /storage/searchfeeds/{search-feed-id} delete a search feed

PUT /storage/searchfeeds/{search-feed-id} update a search feed

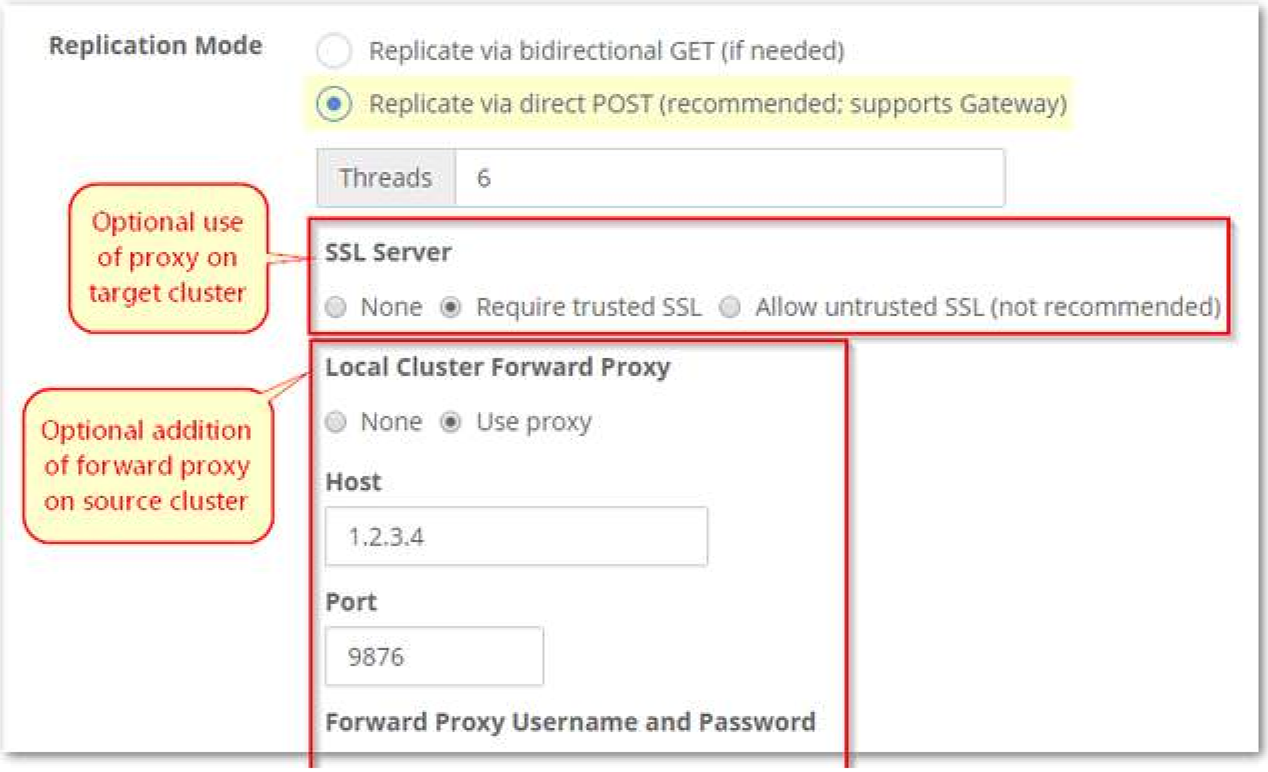
PATCH /storage/searchfeeds/{search-feed-id} update fields of a search feed

Click to open documentation and "Try it out"

- Dashboard
  - The dashboard now dynamically reports the amount of space available as well as space used in the storage cluster. (UIS-765)

Replication Feeds with SSL

With 2.0, Replicate via direct POST now supports SSL/TLS network encryption and standard proxy servers for replication feeds, which eliminates the need for separate VPN tunnels between clusters. This capability streamlines deployments where encrypted communications are needed over wide-area, untrusted networks. See Replicating Feeds over Untrusted Networks.



Replication Mode

○ Replicate via bidirectional GET (if needed)

● Replicate via direct POST (recommended; supports Gateway)

Threads: 6

SSL Server

○ None ● Require trusted SSL ○ Allow untrusted SSL (not recommended)

Local Cluster Forward Proxy

○ None ● Use proxy

Host: 1.2.3.4

Port: 9876

Forward Proxy Username and Password

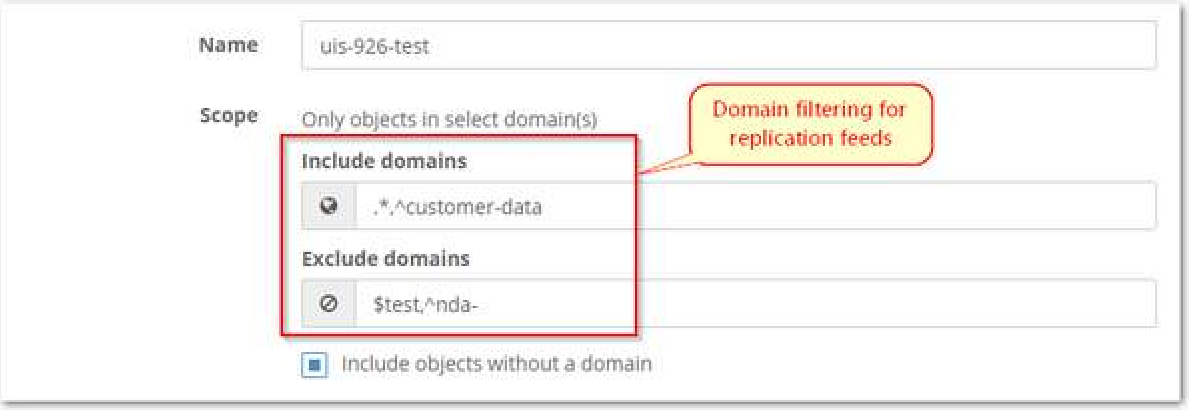
Optional use of proxy on target cluster

Optional addition of forward proxy on source cluster

- Swarm now supports using SSL for remote replication data transfer. This configuration requires an SSL offload proxy (such as HAProxy) in the target cluster environment. (SWAR-7826)
- For sites using SSL with remote replication, Swarm allows the establishment of trusted certificates (public keys) that may be self-signed. (SWAR-8080) See Adding a Trusted Certificate to Swarm.
- Swarm now lets you put a forward proxy to the source cluster into the replication path. (SWAR-8025)

Feed Control and Monitoring

- The statuses reported on the Cluster > Feeds and Reports > Feeds pages now refresh automatically. (UIS-796)
- Feed table — To support feed troubleshooting, the Feed Settings page for a given feed now includes a command to View feed table, which displays the the SNMP repository dump for the selected node. (UIS-787)
- Domain filtering — When defining new replication feeds, you can now filter which domains to include, exclude, or both, and you can specify whether to replicate any unnamed objects that are not tenanted in any domain. The domain filters support wildcard matching for ease of maintenance. (UIS-709)



Name: uis-926-test

Scope: Only objects in select domain(s)

Include domains: .*,^customer-data

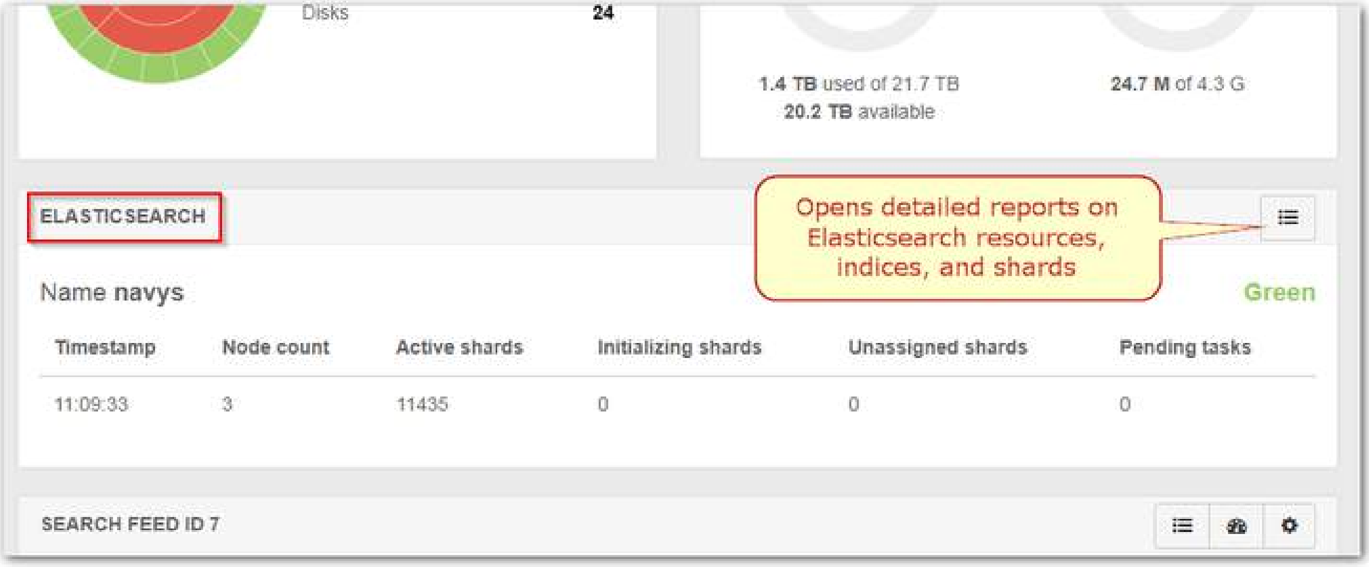
Exclude domains: $test,^nda-

■ Include objects without a domain

Domain filtering for replication feeds

Elasticsearch Cluster Status

If the Elasticsearch panel on the Dashboard shows a problem, you can research your ES cluster status on the Elasticsearch Reports page.

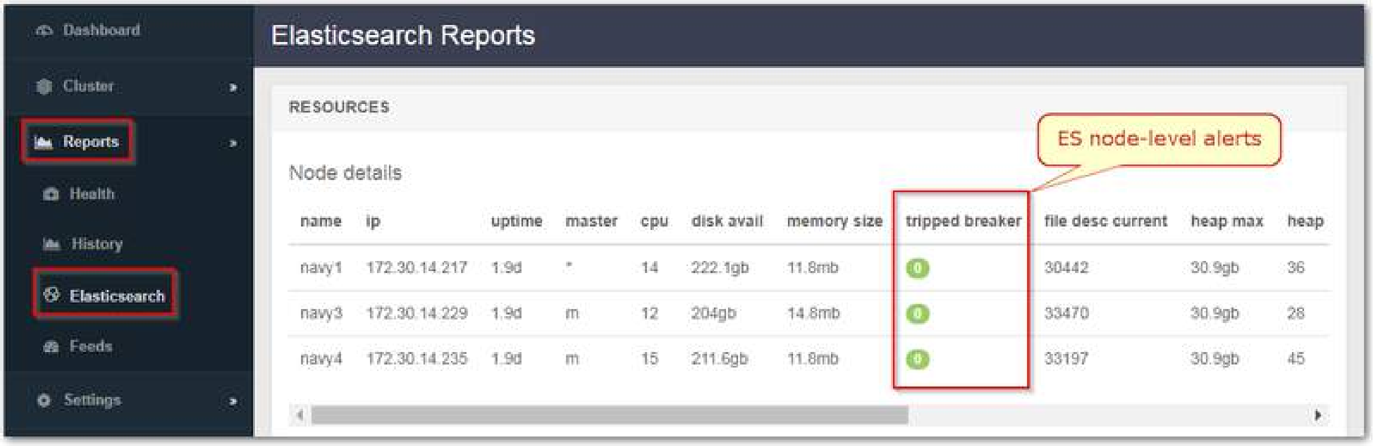


Opens detailed reports on Elasticsearch resources, indices, and shards

Name navys Green

Timestamp	Node count	Active shards	Initializing shards	Unassigned shards	Pending tasks
11:09:33	3	11435	0	0	0

These reports generate on demand and let you drill into details spanning the ES nodes, thread pools, indices, and shards. See Using Cluster Reports.



Elasticsearch Reports

RESOURCES

ES node-level alerts

Node details

name	ip	uptime	master	cpu	disk avail	memory size	tripped breaker	file desc current	heap max	heap
navy1	172.30.14.217	1.9d	*	14	222.1gb	11.8mb	0	30442	30.9gb	36
navy3	172.30.14.229	1.9d	m	12	204gb	14.8mb	0	33470	30.9gb	28
navy4	172.30.14.235	1.9d	m	15	211.6gb	11.8mb	0	33197	30.9gb	45

Additional Changes

- Improved: The cluster-level Health Report now totals stream counts for individual machines (chassis) and for the entire cluster. (UIS-716)
- Improved: The Swarm UI now supports being served on ports other than 91, as governed by the `cluster_admin` bind port you set in the Gateway configuration. This change lets you use the binding that is needed for your environment, such as for a proxy or in a Docker environment. (UIS-934)
- Fixed: Storage UI did not display search feed configuration if the Elasticsearch service was paused. (UIS-766)

Watch Items and Known Issues

- When editing the Rolling Restart Queue, if you deselect nodes from the queue but do not launch the Restart, the UI will reselect them all. (UIS-957)
- When you try to view Health Data (the raw JSON of the health report) on the Advanced tab of the Chassis Details page, the node may become temporarily unresponsive. (SWAR-8349)
- While a reboot of a storage node is in progress, it may be reported to be in an unknown state rather than in maintenance mode. (SWAR-8348)
- On the Details tab of the Chassis Details page, the Actions menu for each disk erroneously shows a testing-only command to Fail the disk. Selecting the command causes errors to display but does not affect your disk (UIS-955)
- Issues exist with feeds that were defined to use a non-default admin password. (UIS-759)
- When using the Swarm UI to identify volumes, turn off the identify function before removing the disk from the chassis. Failure to do so could result in the need to restart the chassis. (UIS-564)
- If a cluster has no search feed defined yet, clicking Add Export on the NFS page causes an immediate 500 error. (UIS-441)
- When you pause a feed, the UI does not convey that the pausing cannot begin until the feed backlog is cleared, which can be a long delay. (UIS-437)

Swarm Platform Release Notes

- Changes in Platform 9.1
- Changes in Platform 9.0

Migrating from CSN

Direct upgrades from CSN 8.3 are not supported. To implement Swarm Platform 9.x, install on a newly provisioned cluster, or contact Support for help with your migration.

Changes in Platform 9.1

This release features these major improvements:

- UEFI Network Booting Support — Platform now includes support for UEFI so that Swarm Storage nodes can be network booted using BIOS or UEFI. As part of this boot support, MAAS is upgraded to version 2.3, which has better support for offline installation and is able to work behind a proxy server, for web access. (ZINC-526, ZINC-495)
    - Upgrade impact: The new default kernel arg only applies to chassis that are deployed after the upgrade. For existing chassis, you must use the existing CLI commands for kernel args. (ZINC-478)
- Rolling Reboot API and CLI Support — Platform now enables you to conduct rolling reboots of active Swarm clusters through its management API. It also includes CLI control over the rolling reboot process with commands to pause, resume, skip specific nodes, and cancel rolling reboots altogether. (ZINC-499)
  See Platform Administration.
- Bonding Mode and Default Gateway — When implementing Platform, you can set and later modify the bonding mode that is used by Swarm Storage (see Deploy Storage, Network Bonding Mode). You can also change the default gateway that is used by Swarm (see Changing the Default Gateway). (ZINC-460)
  See Platform Administration.
- New CLI Commands — For this release, Platform's CLI commands have been extensively expanded and simplified. Following is a complete list of all of the new commands available.
  See Platform CLI Commands.
    - platform add
    - platform add adminuser
    - platform add bonding-mode
    - platform add config
    - platform add kernelparam
    - platform add tag
    - platform delete
    - platform delete adminuser
    - platform delete allchassisconfig
    - platform delete allclusterconfig
    - platform delete bonding-mode
    - platform delete config
    - platform delete kernelparam
    - platform delete proxyimage
    - platform delete rollingreboot
    - platform delete tag
    - platform deploy
    - platform deploy proxy
    - platform list
    - platform list adminusers

  - platform list bonding-mode
  - platform list license
  - platform list proxyimages
  - platform list tags
  - platform pause
  - platform pause rollingreboot
  - platform release
  - platform release proxy
  - platform release storagechassis
  - platform restart
  - platform restart proxy
  - platform restart storagecluster
  - platform restart storagenode
  - platform resume
  - platform resume rollingreboot
  - platform status
  - platform status platform
  - platform status proxy
  - platform status rollingreboot
  - platform status storagechassis
  - platform status storagecluster
  - platform subcluster
  - platform subcluster assign
  - platform subcluster list
  - platform subcluster unassign
- Fixed: The CLI command 'platform list config' failed when run immediately after initial installation. (ZINC-399)

Changes in Platform 9.0

Swarm Platform release 9.0 introduces the next-generation Platform server for new installations of Swarm Storage and Content Gateway.

- Known issue: The CLI command 'platform list config' fails when run immediately after the initial deployment. (ZINC-399)

CSN Platform Server Release Notes

The Swarm CSN Platform Server streamlines installation and configuration of both the network services required to run a Swarm cluster and the Caringo software used to interface with it into a single integrated server.

- Changes by Release
- Limitations and Known Issues
- Upgrading the CSN
- Application and Configuration Notes

Changes by Release

Release 8.3.2

The 8.3.2 package includes a fix for a security hole that affects RHEL/CentOS 6.10. (CSN-2113)

Release 8.3.1

The 8.3.1 package supports upgrading and installing on RHEL/CentOS 6.8 or newer. (CSN-2107, CSN-2112)

If you upgraded the CSN 8.3.0 Swarm UI to RHEL/CentOS 6.9 or newer before installing CSN 8.3.1, be sure to prime the aliases for the metrics indices:

```
/usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n -v
```

Upgrade impact: Customizations made to `castor_net` in the kernel options of the `netboot.cfg` are lost on upgrade and must be restored; other changes (`gateway`, `maxProcesses`) may not be preserved. (CSN-2081)

Release 8.3

> Subclusters
> As of Swarm Storage 9.3.1, subcluster assignments can no longer be blank, and CSN installations with mixed subcluster assignments will have the unassigned nodes unable to boot, showing an error in contacting their time source. Be sure to supply a subcluster for each node if any named subcluster is specified in your cluster. (SWAR-7675)

New Swarm Storage UI: CSN now includes the redesigned storage interface so that you can transition from the legacy Swarm Admin Console while still using the familiar CSN software stack.

- The new UI is available from the URL `http://{CSN·host}:91/_admin/storage` and includes the ability to view, configure, and manage the storage cluster. (CSN-2057)
- When deployed on a dual-network CSN, a new Service Gateway component bridges the communications between the front-side network and the back-end network that connects the storage nodes. See Installing the Storage UI.
- The legacy Swarm Admin Console is still available during the transition to the new UI.
- See Swarm Storage UI in the Swarm Storage guide.

Historical Metrics hosted on CSN: CSN includes all components needed to allow the CSN to collect and store the operational time-series data produced by the storage cluster and used by the new UI. This means that deployments that do not make use of a separate Elasticsearch cluster for metadata searching can now collect these metrics.

- The software components and configuration to host a metrics-only Elasticsearch database are included in the CSN software bundle. This is intended for deployments that do not already have a separate Elasticsearch cluster for metadata searching. (CSN-2060)
- After starting historical metrics collection, it can take a minimum of 30 minutes before the new UI can render

trend graphs based on the data.
- See Swarm Historical Metrics in the Swarm Storage guide.

Other changes:

- The CSN Console (`http://{CSN·host}:8090`) has removed the legacy Reporter functionality, which is replaced by Swarm Metrics in the Swarm Storage UI. (CSN-2058)
- The CSN Console has removed the options for persisted settings, which are no longer configurable from the CSN Console but are settable in the Swarm Storage UI.
- CSN includes updated NTP packages. (CSN-2016)

Release 8.2

- CSN has been updated to install on RHEL/Centos 6.8.
- The CSN software bundle includes a version update to Swarm 8.2.1. Replication configuration settings are automatically migrated to work with new Swarm domain and bucket-level policies as part of the CSN upgrade process. See Configuring Content Policies.
- The CSN software bundle includes a version update to SCSP Proxy 8.2.

Release 8.1

- The CSN software bundle includes a version update to Swarm 8.1. Erasure coding configuration settings are automatically migrated to work with new Swarm domain and bucket-level policies as part of the CSN upgrade process. See Configuring Content Policies.
- Due to breaking changes in the cluster.cfg EC settings for Swarm 8.1, CSN 8.1 and higher prevent you from restoring CSN backups from versions prior to 8.1. (CSN-2025)

Release 8.0

- The CSN software bundle includes a version update to Swarm 8.0. Content Router is no longer included or supported.
- An error in the CSN's support data collection tool, sosreport, has been corrected to ensure all the necessary information is available when filing a support ticket.

Limitations and Known Issues

These are the known issues and operational limitations that exist in this release of Swarm CSN.

- If you install a single-network CSN on a machine that has multiple NICs that are live, it will fail silently. (CSN-2116)
- Changes made to /etc/caringo/netboot/netboot.cfg for the default line of "kernelOptions = castor_net=balance-alb:" will not be preserved through upgrades to CSN. (CSN-2081)
- Upgrading to CentOS 6.8 while the EPEL repo is enabled will upgrade monit to version 5.14.1, which breaks the firstcsnboot script. Do not enable the EPEL repo until after the CSN has been installed. (CSN-2053)

Upgrading the CSN

> Important
>
> Configuration backups taken prior to version 8.2 are not compatible with version 8.2 due to both the end of life of Content Router and settings changes required in Swarm 8.2.

To upgrade a version 3.x or later CSN, you must first upgrade to RHEL or CentOS 6.8 and then install the new software using the included installation script. You will need to re-enable yum updates before upgrading.

> See CSN Updates and Upgrades.

> Note
>
> CSN 7.0.1 was the last release for which upgrade from 2.x was supported. To upgrade from 2.x requires two steps: upgrading first to 7.0.1 and then again to the latest version.

Application and Configuration Notes

These CSN configuration issues require special attention:

- SNMP errors related to _ifXTable_container_row_restore. In some cases, you might see errors similar to the following when installing CSN: error finding row index in _ifXTable_container_row_restore This error is harmless and can be ignored. For more information, see Red Hat bug 736264.
- Messages to Ignore during Installation and Initial Configuration. During the NIC portion of initial configuration, messages similar to 'eth0 NIC Copper Link is Down' may display. After you install and initially configure CSN, service Fail messages might display before CSN reboots. Additionally, there may be logged errors related to an inability to resolve FQDN names. These messages are all harmless having to do with either interim system status as the configuration is applied or timing related states that are retried. The messages can safely be ignored.
- Clean System Start State. The CSN assumes a clean system start state prior to installation, with no previous network configuration of NICs and ports. In particular, manual pre-configuration of NIC aliases (ifcfg-eth1:1) may lead to unexpected network behavior on the CSN as the network configuration performed during installation will not be aware of the pre-configured aliases.
- RedHat Administrative Tool Compatibility. Both the RedHat Network Administration Tool and the RedHat Security Level Configuration Tool are incompatible with the CSN because they alter the expected location and/or configuration of critical network and firewall configuration files, possibly interfering with initial CSN configuration or communications. To prevent conflicts, these tools are removed during installation of the CSN.
- SELinux Errors. While SELinux is run in permissive mode during the bootstrap process, SELinux errors may still appear during bootstrap. These errors can be safely ignored and will not occur again after a system reboot.
- Monit Configuration. The monit watchdog process is carefully configured on the CSN to ensure compatibility with different products and services. Manual configuration of monit parameters is not supported.
- SSH Delays Logging into CSN. Administrators may see delays logging into the CSN via SSH if there are any non-existent or unreachable DNS servers specified in the resolv.conf file. Adding a "UseDNS no" line to the /etc/ssh/sshd_config should remove the delay.

Content Gateway Release Notes

Content Gateway is a lightweight, web-scale application used by companies who want to deploy massively scalable, secure, multi-tenant object storage clouds. Its primary components include a Gateway, a Content Portal, and a Metering service.

> Note
>
> If you are upgrading from a prior version, be sure to review the New Features and Functional Changes for each version since the version from which you are upgrading.

Release Notes for Content Gateway 5.4

New Features

This release includes these improvements and fixes:

- Docker or proxy use — Gateway can now be configured for use either within a Docker environment or behind a proxy. The Gateway configuration has two new settings available (`externalHTTPPort, externalHTTPSPort`) per protocol: [scsp] and [cluster_admin], the Service Proxy. They only take effect when X-Forwarded-Proto is found on the request; Gateway uses X-Forwarded-Proto to determine which port to use. (CLOUD-3055)
- Metadata Translation — Gateway seamlessly moves custom metadata between SCSP and S3. It translates the metadata formatting between the protocols, which means that it now provides S3 and SCSP applications the ability to access each other's metadata. See Metadata Translation between SCSP and S3. (CLOUD-2919)
- Quota stability — Quota implementation has been significantly hardened, resolving general problems with frozen usage status/data and excessive emails from state changes. (CLOUD-3045)
- Fixed issues:
  - There were issues when enabling versioning or quotas through the Content UI. The fix prevents creation of duplicate (conflicting) domains in the storage cluster. (CLOUD-3019)
  - Updating certain Swarm configuration settings through the Gateway services proxy (port 91) sometimes required the sptid query arg value. (CLOUD-2999)
  - Gateway validates on startup that [storage_cluster] managementPassword is set whenever the ServiceProxy ([cluster_admin]) is enabled. (CLOUD-2617)

Upgrade Impacts

- Swarm Storage Requirements — Gateway 5.4 requires a minimum back-end storage cluster version of 9.6. The storage cluster must be upgraded prior to installing this version of Gateway.
  - Releases of Gateway 5.1.3 and earlier are not compatible with Storage versions 9.2 or higher. For historic compatibility details, refer to the releases included with those distributions.

- RHEL/CentOS 6 Deprecation — Support for RHEL/CentOS 6 will be removed in the near future. Completing the transition to RHEL/CentOS 7 now assures a smooth upgrade path to future Gateway versions.
- Automatic service start must be re-enabled — After the upgrade, the service might not automatically start after a system reboot. To re-enable the service, run:
  "`chkconfig --add cloudgateway`" for RHEL/CentOS 6, or
  "`systemctl enable cloudgateway`" for RHEL/CentOS 7. (CLOUD-2819)
- ExpanDrive — ExpanDrive users must upgrade to ExpanDrive 6.1.0 with this version of Gateway; earlier versions report 403 Signature errors when creating a folder or uploading a large object. (CLOUD-2746)

Watch Items and Known Issues

These are known operational limitations and watch items that exist for Gateway.

- When using the default RHEL/CentOS configuration of IPTABLES, traffic to the Gateway will be blocked unless action is taken to disable IPTABLES or to enable inbound traffic to the front-end protocol port(s).
- Gateway is not compatible with the fingerprint scanner module for Linux PAM. If it is installed, remove it by running: `yum remove fprintd-pam`
- SCSP reading operations that request a Content-MD5 hash validation and for which there is a hash mismatch will cause a storage node to be temporarily removed for the Gateway's connection pool due to the way that Swarm reports a hash validation failure.
- Swarm Integrity Seal upgrades cannot be performed through Gateway. They may be done directly to the back-end Swarm cluster.
- If the HTTP cache control headers `If-Modified-Since` and `If-Unmodified-Since` are used, review the discussion of these in the Storage SCSP Development.

The following are known issues in this release.

- The Gateway error "Failed reading from client" on a PUT due to "EofException: Early EOF" might occur when clients do not send the full body. This may point to a bug in the client's retry logic, such as not resetting the position marker to the beginning of the file or part. CLOUD-3010
- During new object creation as part of renaming with ?newname, Gateway does not verify that the user has permission to create the new object name (although it's highly likely, because it's a write within the same context). CLOUD-2966
- An s3cmd or rclone server-side copy request might time out on a multipart copy for >5GB objects (s4cmd performs it correctly). Workaround: After you verify that it's not the HTTPS proxy timing out, increase the client timeout: set s3mcd socket_timeout = 600 in ~/.s3cfg or use rclone copy --timeout=10m --contimeout=2m caringo:mybucket/5gb caringo:mybucket/subfolder/. CLOUD-2949
- Listings with max-keys might be shorter than expected because CommonPrefixes are included in the count of keys returned. CLOUD-2917
- Uploading files / photos using Panic's Transmit app on iOS fails due to a 403 Invalid Signature error. CLOUD-2886
- Gateway 5.2.2 and earlier do not output the NextMarker field in S3 listings, which can cause some S3 clients such as Caringo Drive, rclone, and Transmit to show only 1000 files in a directory or to miss some subdirectories. CLOUD-2871
- Usernames are case-insensitive, but listings will exclude a token if the username (myadmin) does not match the case used when the token was created (myAdmin). CLOUD-2837
- Upgrade impact: After an upgrade from version 5.2.1 or earlier, the Gateway service will not automatically start after a system reboot. To re-enable the service, run "chkconfig --add cloudgateway" (RHEL 6) or "systemctl

enable cloudgateway" (RHEL 7). CLOUD-2819

- Multipart PUT requests via recent Cyberduck versions fail with 403 SignatureDoesNotMatch when using AWS Signature Version 4. Please install the Caringo .cyberduckprofiles from https://caringo.atlassian.net/servicedesk/customer/kb/view/37134679 which force V2 signatures. CLOUD-2799
- If a policy document includes a Principal that has plural "users" or "groups" instead of "user" or "group", the policy will fail to take effect without warning. CLOUD-2783
- Versioning-enabled buckets with large numbers of objects may generate Gateway server.log warnings that can be safely ignored: "S3BucketRequestHandler: WARNING: problem with versioned bucket listing. Number of CommonPrefix (2000) exceeds max-size limit (1000)." CLOUD-2643
- 403 S3 V4 Signature mismatch errors may result when using Cyberduck with the "pound" proxy in front of Gateway S3. Workaround: Disable the Expect header in the Cyberduck preferences, or (recommended) use a different proxy such as "haproxy". CLOUD-2628
- When Gateway cannot connect to Elasticsearch nodes, the errors may erroneously report this as being related to Storage nodes. CLOUD-2595
- Because of issues with Range and ETag header handling, video playback of .mp4 streams might not work correctly when served via the Gateway S3 port. It does work when served via the Gateway SCSP port. CLOUD-1964
- Gateway caches the Swarm version from the "Server:" response header, so after upgrading Swarm you must restart Gateway to consistently see the new version. CLOUD-1271
- Gateway responds with a 500 (Internal Server Error) instead of 400 (Bad Request) if the size of the metadata headers sent to Swarm is too large. CLOUD-800
- The S3 bucket listing StorageClass response element always reports STANDARD. CLOUD-766
- If an S3 client escapes URI path characters such as "/", the Gateway audit log will escape the "%" characters used by the client as escape characters. URI audit log processing for S3 clients will require double-unescaping when this occurs. CLOUD-703

Upgrading from Release 5.x

The Gateway software components packaged as RPMs in the distribution and this section describes the method to upgrade existing 5.x installations. Although these steps include instructions for routing traffic away from Gateways during the upgrade process, you may safely skip these if your deployment does not include a load balancer.

1. Ensure Swarm storage cluster is upgraded to version 9 or later. This includes completing the migration to Elasticsearch 2.3 metadata search servers in conjunction with the Swarm 9 upgrade.
2. In your load balancer, disable traffic for one Gateway at a time and perform the following upgrade steps for that Gateway. Allow traffic to continue flowing to the other Gateways.
3. Upgrade the traffic-disabled Gateway.
    a. Stop the Gateway service.

RHEL/CentOS 6

```
service cloudgateway stop
```

RHEL/CentOS 7

```
systemctl stop cloudgateway
```

b. Upgrade the Gateway service:

```
yum -y install caringo-gateway-{version}.rpm
```

c. If using RHEL/CentOS 7, reload the systemd control scripts:

```
systemctl daemon-reload
```

d. Review and merge any configuration changes to the `gateway.cfg` and `logging.cfg` files in the `/etc/caringo/cloudgateway` directory. The new distribution versions are the `*.rpmnew` files.

e. Start the Gateway service:

RHEL/CentOS 6

```
service cloudgateway start
```

RHEL/CentOS 7

```
systemctl start cloudgateway
```

f. Confirm that the Gateway service is running. See the log and curl examples in "Verifying the upgrade," below.

g. Upgrade the Content Portal:

```
yum -y install caringo-gateway-webui-{version}.rpm
```

4. In your load balancer, re-enable client traffic to the newly upgraded Gateway and repeat the process for your remaining Gateways.

Upgrading from Release 4.x

The Gateway software components packaged as RPMs in the distribution and this section describes the method to upgrade existing 4.x installations. Although these steps include instructions for routing traffic away from Gateways during the upgrade process, you may safely skip these if your deployment does not include a load balancer. Since Gateway 4.x is only certified on RHEL/CentOS 6, the commands shown here are only applicable for RHEL/CentOS 6.

1. Ensure Swarm storage cluster is upgraded to version 9 or later. This includes completing the migration to Elasticsearch 2.3 metadata search servers in conjunction with the Swarm 9 upgrade.
2. In your load balancer, disable traffic for one Gateway at a time and perform the following upgrade steps for that Gateway. Allow traffic to continue flowing to the other Gateways.
3. Upgrade the traffic-disabled Gateway.
    a. Stop the Gateway service:

    ```
    service cloudgateway stop
    ```

    b. Upgrade the Gateway service:

    ```
    yum -y install caringo-gateway-{version}.rpm
    ```

    c. Review and merge any configuration changes to the `gateway.cfg` and `logging.cfg` files in the `/etc/caringo/cloudgateway` directory. The new distribution versions are the `*.rpmnew` files.
    d. Start the Gateway service:

    ```
    service cloudgateway start
    ```

    e. Confirm that the Gateway service is running. See the log and curl examples in "Verifying the upgrade," below.
    f. Upgrade the Content Portal:

    ```
    yum -y install caringo-gateway-webui-{version}.rpm
    ```

4. In your load balancer, re-enable client traffic to the newly upgraded Gateway and repeat the process for your remaining Gateways.

Verifying the upgrade

After upgrading, you can check the Gateway's server log (default: `/var/log/caringo/cloudgateway_server.log`) to confirm there are no startup errors. You are looking to confirm that the SCSP and/or S3 protocol services are running. This is an example from a server with both protocols enabled:

```
2015-09-30 11:01:10,382 INFO [main|0000000000000000] S3GatewayServlet:
S3 API enabled
2015-09-30 11:01:10,382 INFO [main|0000000000000000] ScspGatewayServlet:
SCSP API enabled
```

You can also confirm that the Gateway is responding to Management API requests by requesting the API version. This command is run from the Gateway and assumes the SCSP protocol is running on port 8080. Adjust the port and/or interface IP address if your configuration is different.

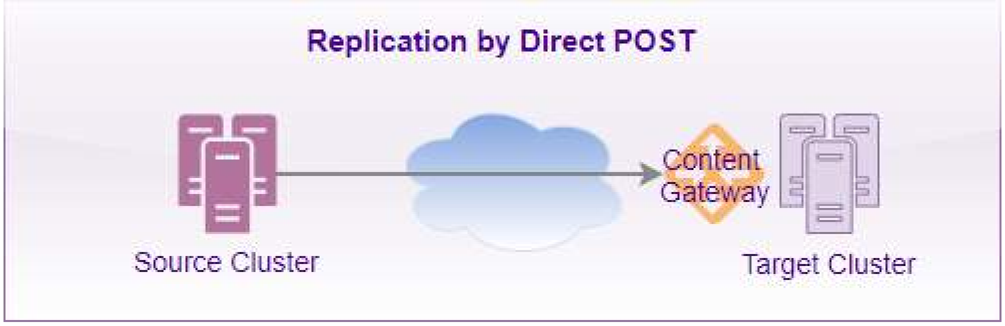
```
curl -i http://localhost:8080/_admin/manage/version
```

You should get an HTTP 200 response and a JSON body returned with that request.

Release Notes for Content Gateway 5.3

New Features

With Swarm Storage 9.6 and the new direct POST method of replication available when defining replication feeds, Content Gateway now supports replication to a remote target cluster. Note that use of SCSP Proxy is no longer required to implement remote replication. (CLOUD-2618)



See Viewing and Editing Feeds.

This release also includes these improvements:

- Gateway has S3 protocol enhancements in order to track with Amazon S3 changes,.
- Gateway has more support for Swarm Platform Server and third-party clients.
- Gateway has better handling for requests to domains that are marked as belonging to an unknown tenant. These requests are handled as if they are from the "System Tenant" and troubleshooting guidance is recorded in the

application logs. (CLOUD-2714)

Upgrade Impacts

- Swarm Storage Requirements — Gateway 5.3 requires a minimum back-end storage cluster version of 9.5.3 and a recommended version of 9.6. The storage cluster must be upgraded prior to installing this version of Gateway.
  - Releases of Gateway 5.1.3 and earlier are not compatible with Storage versions 9.2 or higher. For historic compatibility details, refer to the releases included with those distributions.
- RHEL/CentOS 6 Deprecation — Support for RHEL/CentOS 6 will be removed in the near future. Customers are encouraged to complete the transition to RHEL/CentOS 7 now in order to assure a smooth upgrade path to future Gateway versions.
- Automatic service start must be re-enabled — After the upgrade, the service might not automatically start after a system reboot. To re-enable the service, run:
  "`chkconfig --add cloudgateway`" for RHEL/CentOS 6, or
  "`systemctl enable cloudgateway`" for RHEL/CentOS 7. (CLOUD-2819)
- ExpanDrive — ExpanDrive users must upgrade to ExpanDrive 6.1.0 with this version of Gateway; earlier versions report 403 Signature errors when creating a folder or uploading a large object. (CLOUD-2746)

Additional Changes

- Fixed: The cloudgateway_server.log had invalid SCSP warnings reporting 'Failed requests will not be retried' and 'Failed querying cluster for name and version'. (5.3.3: CLOUD-2663)
- Fixed: During S3 multipart "complete" operations, whitespace ticks (meant to keep the connection alive) were not being sent, which caused some clients and load balancers to time out for large number of parts. (5.3.3: CLOUD-3000)
- Fixed: During S3 multipart "complete" operations, an error condition could cause future uploads to report "java.lang.IllegalStateException: Timer already cancelled", although the object usually did complete in Swarm. (5.3.3: CLOUD-2985)
- Fixed: S3 multipart listings returned the incorrect error "part-number-marker must be an integer between 0 and 10000". (5.3.3: CLOUD-2971)
- Fixed: Long S3 multipart "complete" requests could lead to failed client retries. (5.3.2: CLOUD-2736)
- Fixed: S3 multipart listings returned the incorrect error "part-number-marker must be an integer between 0 and 10000". (5.3.1: CLOUD-2937)
- Fixed: 405 errors occurred with CatDV and the Java AWS SDK TransferManager. (CLOUD-2921)
- Fixed: Cyberduck operations like "Edit with" and uploads that "Overwrite?" could fail with a 400 "CAStor Error" / "Content-MD5 did not match" because Content-MD5 was included in GET/HEAD responses, which caused problems with specific clients that failed to strip it from subsequent writes. (CLOUD-2914)
- Fixed: Multipart/form-data MIME POST uploads that spanned several days of sustained load could result in an OutOfMemoryError. (CLOUD-2732)
- Fixed: Buckets did not always inherit versioning status correctly. (CLOUD-2577)
- Fixed: The Date header was missing from the responses for S3 Upload Part operations. (CLOUD-2570)
- Implementing Content Gateway 5.3

Implementing Content Gateway 5.3

Watch Items and Known Issues

These are known operational limitations and watch items that exist in this release of Gateway.

- When using the default RHEL/CentOS configuration of IPTABLES, traffic to the Gateway will be blocked unless action is taken to disable IPTABLES or to enable inbound traffic to the front-end protocol port(s).
- Gateway is not compatible with the fingerprint scanner module for Linux PAM. If it is installed, remove it by running: `yum remove fprintd-pam`
- SCSP reading operations that request a Content-MD5 hash validation and for which there is a hash mismatch will cause a storage node to be temporarily removed for the Gateway's connection pool due to the way that Swarm reports a hash validation failure.
- Swarm Integrity Seal upgrades cannot be performed through Gateway. They may be done directly to the back-end Swarm cluster.
- If the HTTP cache control headers `If-Modified-Since` and `If-Unmodified-Since` are used, review the discussion of these in the Storage SCSP Development.

The following are known issues in this release.

- During new object creation as part of renaming with ?newname, Gateway does not verify that the user has permission to create the new object name (although it's highly likely, because it's a write within the same context). (CLOUD-2966)
- Verbose log4j messages are printed on the console during service start. These have no operational impact and may be ignored. (CLOUD-2964)
- Listings with max-keys might be shorter than expected because CommonPrefixes are included in the count of keys returned. (CLOUD-2917)
- S3 listings that specify a delimiter can return multiple CommonPrefixes entries. That confuses some S3 clients like Caringo Drive, leading to missing directories. (CLOUD-2888)
- Uploading files/photos using Panic's Transmit app on iOS fails due to a 403 Invalid Signature error. (CLOUD-2886)
- Gateway 5.2.2 and earlier do not output the NextMarker field in S3 listings, which can cause some S3 clients such as Caringo Drive, rclone, and Transmit to show only 1000 files in a directory or to miss some subdirectories. (CLOUD-2871)
- Listings will exclude a token if the case of the username (myadmin) does not match that used when the token was created (myAdmin). (CLOUD-2837)
- Multipart PUT requests via recent Cyberduck versions fail with 403 SignatureDoesNotMatch when using AWS Signature Version 4. Please install the Caringo .cyberduckprofiles from https://caringo.atlassian.net/servicedesk/customer/kb/view/37134679 which force V2 signatures. (CLOUD-2799)
- If a policy document includes a Principal that has plural "users" or "groups" instead of "user" or "group", the policy will fail to take effect without warning. (CLOUD-2783)
- On macOS, you must use Transmit 4.x or >=5.0.5 and Gateway 5.2.2 to avoid 403 Invalid Signature errors. (CLOUD-2777)
- Versioning-enabled buckets with large numbers of objects may generate Gateway server.log warnings that can be safely ignored: "S3BucketRequestHandler: WARNING: problem with versioned bucket listing. Number of CommonPrefix (2000) exceeds max-size limit (1000)." (CLOUD-2643)
- 403 S3 V4 Signature mismatch errors may result when using Cyberduck with the "pound" proxy in front of Gateway S3. Workaround: Disable the Expect header in the Cyberduck preferences, or (recommended) use a different proxy, such as "haproxy". (CLOUD-2628)

- When Gateway cannot connect to Elasticsearch nodes, the errors may erroneously report this as being related to Storage nodes. (CLOUD-2595)
- Because of issues with Range and ETag header handling, video playback of .mp4 streams might not work correctly when served via the Gateway S3 port. It does work when served via the Gateway SCSP port. (CLOUD-1964)
- Gateway caches the Swarm version from the "Server:" response header; therefore, after upgrading Swarm, you must restart Gateway to consistently see the new version. (CLOUD-1271)
- Gateway install will fail if Internet access in not available. (CLOUD-1216)
- Gateway responds with a 500 (Internal Server Error) instead of 400 (Bad Request) if the size of the metadata headers sent to Swarm is too large. (CLOUD-800)
- The S3 bucket listing StorageClass response element always reports STANDARD. (CLOUD-766)
- If an S3 client escapes URI path characters such as "/", the Gateway audit log will escape the "%" characters used by the client as escape characters. URI audit log processing for S3 clients will require double-unescaping when this occurs. (CLOUD-703)

Upgrading from Release 5.x

The Gateway software components packaged as RPMs in the distribution and this section describes the method to upgrade existing 5.x installations. Although these steps include instructions for routing traffic away from Gateways during the upgrade process, you may safely skip these if your deployment does not include a load balancer.

1. Ensure Swarm storage cluster is upgraded to version 9 or later. This includes completing the migration to Elasticsearch 2.3 metadata search servers in conjunction with the Swarm 9 upgrade.
2. In your load balancer, disable traffic for one Gateway at a time and perform the following upgrade steps for that Gateway. Allow traffic to continue flowing to the other Gateways.
3. Upgrade the traffic-disabled Gateway.
    a. Stop the Gateway service.

    RHEL/CentOS 6

    ```
    service cloudgateway stop
    ```

    RHEL/CentOS 7

    ```
    systemctl stop cloudgateway
    ```

    b. Upgrade the Gateway service:

    ```
    yum -y install caringo-gateway-{version}.rpm
    ```

    c. If using RHEL/CentOS 7, reload the systemd control scripts:

```
systemctl daemon-reload
```

d. Review and merge any configuration changes to the `gateway.cfg` and `logging.cfg` files in the `/etc/caringo/cloudgateway` directory. The new distribution versions are the `*.rpmnew` files.

e. Start the Gateway service:

RHEL/CentOS 6

```
service cloudgateway start
```

RHEL/CentOS 7

```
systemctl start cloudgateway
```

f. Confirm that the Gateway service is running. See the log and curl examples in "Verifying the upgrade," below.

g. Upgrade the Content Portal:

```
yum -y install caringo-gateway-webui-{version}.rpm
```

4. In your load balancer, re-enable client traffic to the newly upgraded Gateway and repeat the process for your remaining Gateways.

Upgrading from Release 4.x

The Gateway software components packaged as RPMs in the distribution and this section describes the method to upgrade existing 4.x installations. Although these steps include instructions for routing traffic away from Gateways during the upgrade process, you may safely skip these if your deployment does not include a load balancer. Since Gateway 4.x is only certified on RHEL/CentOS 6, the commands shown here are only applicable for RHEL/CentOS 6.

1. Ensure Swarm storage cluster is upgraded to version 9 or later. This includes completing the migration to Elasticsearch 2.3 metadata search servers in conjunction with the Swarm 9 upgrade.
2. In your load balancer, disable traffic for one Gateway at a time and perform the following upgrade steps for that Gateway. Allow traffic to continue flowing to the other Gateways.
3. Upgrade the traffic-disabled Gateway.

   a. Stop the Gateway service:

```
service cloudgateway stop
```

b. Upgrade the Gateway service:

```
yum -y install caringo-gateway-{version}.rpm
```

c. Review and merge any configuration changes to the `gateway.cfg` and `logging.cfg` files in the `/etc/caringo/cloudgateway` directory. The new distribution versions are the `*.rpmnew` files.
d. Start the Gateway service:

```
service cloudgateway start
```

e. Confirm that the Gateway service is running. See the log and curl examples in "Verifying the upgrade," below.
f. Upgrade the Content Portal:

```
yum -y install caringo-gateway-webui-{version}.rpm
```

4. In your load balancer, re-enable client traffic to the newly upgraded Gateway and repeat the process for your remaining Gateways.

Verifying the upgrade

After upgrading, you can check the Gateway's server log (default: `/var/log/caringo/cloudgateway_server.log`) to confirm there are no startup errors. You are looking to confirm that the SCSP and/or S3 protocol services are running. This is an example from a server with both protocols enabled:

```
2015-09-30 11:01:10,382 INFO [main|0000000000000000] S3GatewayServlet:
S3 API enabled
2015-09-30 11:01:10,382 INFO [main|0000000000000000] ScspGatewayServlet:
SCSP API enabled
```

You can also confirm that the Gateway is responding to Management API requests by requesting the API version. This command is run from the Gateway and assumes the SCSP protocol is running on port 8080. Adjust the port and/or interface IP address if your configuration is different.

```
curl -i http://localhost:8080/_admin/manage/version
```

You should get an HTTP 200 response and a JSON body returned with that request.

Release Notes for Content Gateway 5.2

- New Features
- Upgrade Impacts
- Additional Changes

New Features

This release focuses on improvements to support Amazon S3 compatibility. Specifically and to match Amazon S3 behavior:

- Gateway utilizes a new feature in Swarm storage 9.3 to improve the interoperability with S3 client applications that are not written to properly handle the eventual consistency nature of S3. In deployments where the applications correctly handle the S3 eventual consistency, this feature can be disabled to achieve maximum write throughput on small objects. See the enhancedListingConsistency configuration setting for details. (5.2.1: CLOUD-2734)
- Gateway now responds to a nonexistent access key with a "403 Forbidden" rather than a "400 Bad Request" code. (CLOUD-2698)
- Gateway now responds to a deleted resource with a "204 No Content" rather than a "404 Not Found" code. (CLOUD-2697)
- Gateway changed the handling of a request header "Content-type" containing a lowercase "charset" to prevent 403 signature mismatch errors occurring with some S3 clients (such as rclone) when using S3 V4 signatures. (CLOUD-2664)

Upgrade Impacts

- Swarm Storage 9.3+ Required — Content Gateway 5.2.3 supports Storage 9.3 and newer. The storage cluster must be upgraded prior to installing this version of Gateway. If you are using Storage 9.4, be sure to update to Storage UI 1.2.3, which corrects an issue with defining a replication feed that specifies domains. (CLOUD-2890)
  - Gateway 5.2.2 requires Swarm Storage 9.3+.
  - Gateway 5.2.0 requires Swarm Storage 9.2.1 and is not compatible with Swarm Storage 9.3.
  - Gateway 5.1.3 and earlier are not compatible with Swarm Storage 9.2 and higher. For additional historic compatibility details, refer to the releases included with those distributions.
- EC required — Erasure coding (EC) must be enabled and configured; if not, the Gateway cannot start (`service cloudgateway init` will fail). (CLOUD-2587)
- Elasticsearch 2.3.3 — Swarm 9 includes significant enhancements to its use of Elasticsearch and Gateway now depends upon features from Elasticsearch 2.3. Prior to upgrading Gateway, you must complete the migration to Elasticsearch 2.3.
- RPM changes — The Content Gateway RPM package has changed from "caringo-cloudgateway" to "caringo-gateway". (CLOUD-2728)

- Re-enable service — After an upgrade from version 5.2.1 or earlier, the service will not automatically start after a system reboot. To re-enable the service, run "chkconfig --add cloudgateway" (RHEL 6) or "systemctl enable cloudgateway" (RHEL 7). (5.2.2: CLOUD-2819)
- ExpanDrive — ExpanDrive users must upgrade to ExpanDrive 6.1.0 and Gateway 5.2.2; earlier versions report 403 Signature errors when creating a folder or uploading a large object. (5.2.2: CLOUD-2746)

Additional Changes

- Improved: A new [gateway] setting, cookieDomains, allows the Content Portal to use the same authentication token across multiple storage domains that share a common base domain. Gateway does this by using the base domain in place of the request's domain for the Set-Cookie response header. (5.2.2: CLOUD-2789)
- Improved: Gateway has better handling for requests to domains that are marked as belonging to an unknown tenant. These requests are handled as if they are from the "System Tenant" and troubleshooting guidance is recorded in the application logs. (5.2.1: CLOUD-2714)
- Improved: Gateway has lowered its default request rewind buffer to 64KB in order to prevent an excessive volume of messages in the storage cluster logs and to match the storage cluster's upper threshold after which 100-continue must be used. This should have no impact for applications using Gateway. (CLOUD-2652)
- Fixed: A versioned listing now responds in the correct format when the bucket is not versioned, fixing a problem that occurred with Cyberduck 6.2.3 and later. (5.2.2: CLOUD-2796)
- Fixed: When using a "pam" idsys for Linux users on the Gateway server, requests might slow down after days of high, sustained load. (5.2.1: CLOUD-2727)
- Fixed: S3 V4 signature errors occurred when using X-Forwarded-Host on a request. Setting [debug] setForwarded=false prevents these errors, if they occur. (5.2.1: CLOUD-2706)
- Fixed: Multipart/form-data MIME POST uploads that spanned several days of sustained load could result in an OutOfMemoryError. (5.2.1: CLOUD-2732)
- Fixed: Gateway startup could silently hang if there was a problem communicating with the indexerHosts or if Swarm was not yet configured with a Search feed. (5.2.1: CLOUD-2419)
- Fixed: Gateway log warning messages about "Swarm connection failure to node 127.0.0.1" could be generated when using Swarm UI. (5.2.1: CLOUD-2749)
- Fixed: Single-shot context creation failed in Gateway 5.1.2 and 5.1.3; the POST failed with a 500 "JSON Validation Errors". (CLOUD-2703)
- Fixed: In the Content Portal, a collection could appear as a bucket, such that deleting the collection could delete the bucket that it was scoped to. For any collections that erroneously still show the bucket icon, make a small edit and resave them, which causes them to be fully repaired. (CLOUD-2694, CLOUD-2712)

Release Notes for Content Gateway 5.1

- New Features
- Upgrade Impacts
- Additional Changes

New Features

- Quota Policies - You can now define comprehensive quota policies to enforce usage limits for specific tenants, storage domains, and buckets. Through the Content Portal, you go to the Properties of a tenant, domain, or bucket and set limits for storage usage and/or network bandwidth metrics, selecting which action to occur when

the limit is exceeded and who to notify. The actions (consequences) for overages range from notification only, read and delete only, read only, to complete lockout of access. Both storage and bandwidth limits include override options, which let you create temporary grace periods with lesser restrictions. Setup and customization of quotas occurs through the new [quota] section of the Gateway Configuration.
See Setting Quotas.

- CloudScaler Renamed - The product known as CloudScaler is now referred to as the Content Gateway (inclusive of the Content Portal and Metering). The name change reflects the current and continuing integration of all products into Swarm 9. Note the new RPM names: `caringo-gateway-{version}.rpm, caringo-gateway-webui-{version}.rpm`
- RHEL/CentOS 7 Support - This release is tested and certified with RHEL/CentOS 7. While you can continue to use RHEL/CentOS 6 for Content Gateway, it is recommended that customers migrate to the version 7 platform as soon as practical.
- Viewing Versioned Objects - When multiple versions of an object exist, the Content Portal provides a drop-down list so that the metadata details for each version can be viewed.
- Service Proxy - The new Service Proxy (`ServiceProxyServlet`) works as a special instance of Gateway that you configure with additions and overrides to its configuration. The Service Proxy provides a proxy to a Swarm cluster, leveraging the same IDSYS authorization and authentication as your Content Gateway. With the Service Proxy, you can host the Swarm Storage UI and API from a server that is accessible to your administrators and have the Service Proxy manage communication with the Swarm cluster. Doing so gives you a single access point for managing and monitoring an entire Swarm cluster. (5.1.2: CLOUD-2602)
See Service Proxy.

Upgrade Impacts

- Swarm Storage 9.1.2 - Swarm 9.1.2 is the minimum required version of the back-end storage cluster. The storage cluster must be upgraded prior to installing this version of Gateway.
- Elasticsearch 2.3 - Swarm 9 includes significant enhancements to its use of Elasticsearch and Gateway now depends upon features from Elasticsearch 2.3. Prior to upgrading Gateway, you must complete the migration to Elasticsearch 2.3. See the Swarm 9 storage cluster release notes for details regarding this migration.
- Upon encountering a root IDSYS with an invalid syntax, the HTTP response code from Content Gateway has been changed to 503. It previously returned 403 (Unauthorized) which could lead to confusion with debugging configuration issues.

Additional Changes

- Corrected an issue with S3 multipart writes on storage clusters that had never had a parallel upload performed. This resulted in listing errors when using Cyberduck version 5.0. (5.1.0: CLOUD-2398)
- Case-insensitive metadata searching within Swarm 9 storage cluster is now compatible with the S3 protocol front-end once you complete the migration to Elasticsearch 2.3. (SWAR-5283, SWAR-6317)
- Long metadata field names are truncated in collections display in order to improve readability. (5.1.0: CLOUD-1691)
- Help links for both 'Documentation' and 'Online Support' in the Content Portal are now customizeable through edits to the file `_admin/portal/customLinks.json`. (5.1.0: CLOUD-1987)
- Fixed an issue that caused a failure when using SCSP APPEND through the Gateway. (5.1.2: CLOUD-2629)
- Fixed an issue in which the loss of one Elasticsearch node in a multi-node Elasticsearch cluster resulted in failure to perform queries and perform S3 listings. (5.1.2: CLOUD-2412)

- Fixed an issue that threw an error and prevented an object delete when the context disallowed versioning. (5.1.1: CLOUD-2611)
- Improved the reliability of S3 multipart upload completion. (5.1.2: CLOUD-2414)
- Improved the handling of Swarm errors during S3 multipart upload completion. (5.1.3: CLOUD-2633)
- Fixed: Using the utility 's3cmd' v1.6.1, attempts to perform 's3cmd mv' from one filename to another and/or one bucket to another failed for large (100+ MB) files. (5.1.3: CLOUD-2659)
- Fixed: A change in Swarm 9.1.0 caused S3 header "x-amz-delete-marker: true" to no longer be sent in response to a HEAD of a deleted object, although the header was still returned on GET. (5.1.3: CLOUD-1041)
- Fixed: Gateway 5.0.0 - 5.1.2 S3 listings did not include any ETag for objects uploaded via SCSP parallel write or S3 multipart APIs because they had no Content-MD5. The missing XML key broke clients such as s3cmd. Gateway now includes the same generated (non-MD5) ETag in listings as is returned on a GET of the object. Gateway and Swarm Storage now generate a "composite MD5" of the parts, similar to AWS S3, and Gateway uses that as the object's ETag in S3 response headers and listings. (5.1.3: CLOUD-1496)
- Fixed: Using V4 signatures with some S3 clients (such as rclone) might get a 403 signature mismatch error caused by the request header "Content-type" containing a lowercase "charset". (5.1.3: CLOUD-2664)
- Fixed: When using the S3 list multipart uploads operation, if there are multiple simultaneous uploads in progress for a single object, only one of the uploads will be included in the listing. (5.1.3: CLOUD-2333)

Release Notes for Content Gateway 5.0

- New Features
- Functional Changes
- Additional Changes

New Features

- Redesigned Content Portal - The Content Portal has significant changes in appearance and behavior to support future expansion and integrations. Contents listings appear consistently throughout the Portal, with resizeable and sortable columns, and they have streamlined and simplified commands. (CLOUD-2172, CLOUD-2300)
  See Content UI Overview.
- Content Protection Policies - The Content Portal now allows you to set content protection policies as part of the Properties of domains and buckets, in addition to object versioning. The content protection policies include both replication (how many object copies to maintain in the cluster) and erasure coding (dividing very large objects into data and parity segments, for efficient storage and protection). (CLOUD-1910, CLOUD-2147)
  See Setting Storage Policies.
- Go to Location - The Content Portal now offers a Go To Location feature, which gives you a way to navigate directly to a tenant, domain, or bucket by name. The feature is global, available from all pages as an option on the login menu. The context you select opens to a listing view of its contents. If the requested location is invalid or permission is denied, the error alert gives details and allows you to correct the location. (CLOUD-1739, CLOUD-2302)
  See Go to Location.
- Blocking Undeleteable Objects - Gateway has a new `storage_cluster` configuration option, `blockUndeletableWrites`, that rejects SCSP write requests that include a `deletable=no` lifepoint. This applies to named and unnamed object types and can be useful in storage provider deployments where the cluster administrators wish to retain the final authority to remove content. (CLOUD-1052, CLOUD-2202)
  See Gateway Configuration.

Functional Changes

- Swarm 8.2 is the minimum required version of the back-end object storage engine. Swarm must be upgraded prior to installing this version of Gateway.
- Elasticsearch 1.7.1-7 is the latest version of the metadata search software, and it is included with Swarm 8.2. Version 1.7.1-5 is still supported.
- CSMeter is deprecated and applications should transition to the new metering mechanism. There are no additional updates planned to version 1.1.6 and it is no longer distributed.

Additional Changes

- Parallel Write by Copy - Gateway now supports parallel writes using Swarm's `x-castor-copy-source-*` headers, to upload a part by copying content from an existing object. When making a "PUT with copy" request, the user must have read access to the source. (CLOUD-1920)
- S3 Anonymous Principals - Gateway supports the new AWS S3 format for specifying an anonymous principal. See Policy Document. (CLOUD-1692)
- S3 Empty PUT - For compatibility with AWS S3, Gateway permits PUT without a body and `Content-Length` header, such as for creating a bucket. (CLOUD-2154)
- Performance tuning parameters - The default values for the Java operational parameters to control memory heap and maximum file descriptors have all been increased and are parameterized in the `/etc/sysconfig` directory. See Gateway Operations. (CLOUD-2133, CLOUD-2123)
- Linux PAM user names and group names are now treated as case-sensitive. (CLOUD-2135)
- Added support for multipart uploads that use non-ASCII, UTF-8 object names. (CLOUD-801)
- Improved handling for non-standard bucket names, such as those containing non-alphanumeric characters. (CLOUD-2090)
- Improved cross-domain login handling for cases of failed and restored permissions. (CLOUD-2061)
- Improved tolerance for missing parameters and configuration files on startup. (CLOUD-2168)

Content UI Release

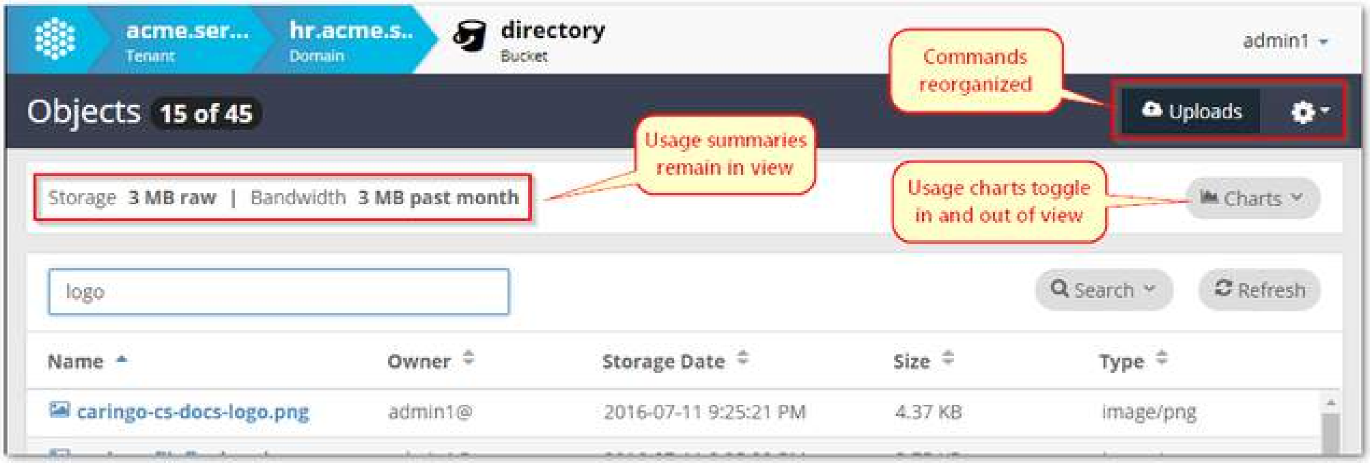
The Swarm Content UI is Gateway's cloud interface to Swarm-based content, providing end users with direct browser access to their content. The Content UI simplifies content management (such as configuring tenants and storage domains and creating search collections based on custom metadata tags), and it also enables end users to download and upload content directly to and from their file systems.

- Changes in Content UI 6.0

Changes in Content UI 6.0

In support of Swarm 10, Content UI has extensive usability upgrades in response to customer feedback. These are highlights:

Streamlined Usage Reports — Page layout and navigation across the Content UI have been reorganized to be more efficient. Because of their importance, Storage and Bandwidth usage summaries were moved into the title bar of the Charts panel, so that they remain in view even when collapsed. Clicking Charts expands and collapses the full view of the charts:

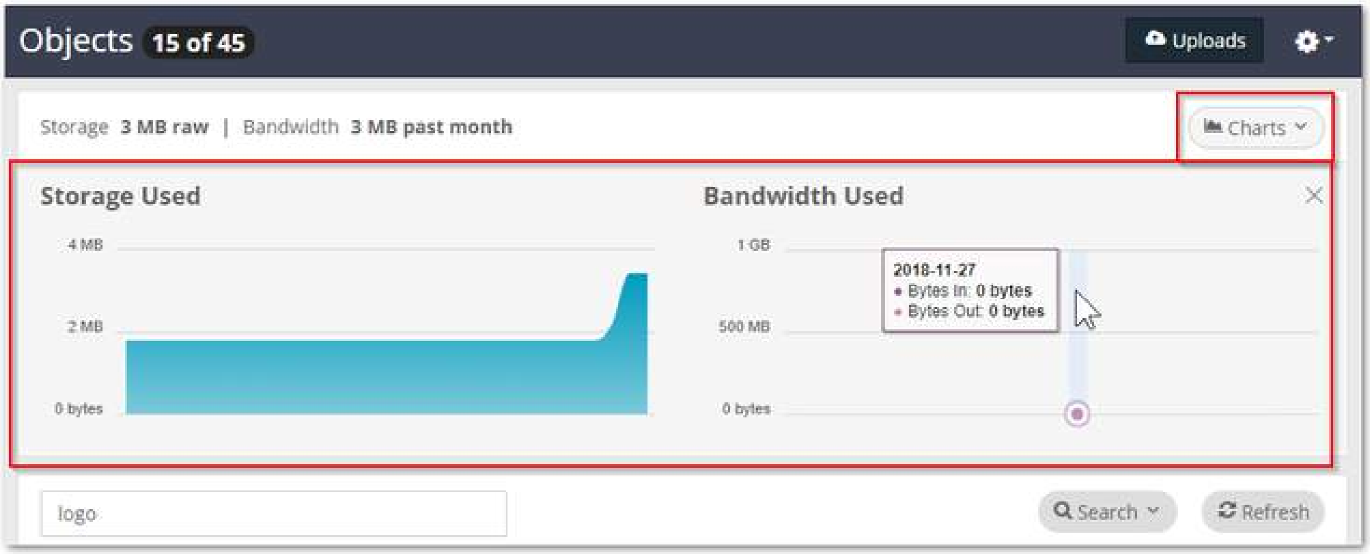


Commands reorganized

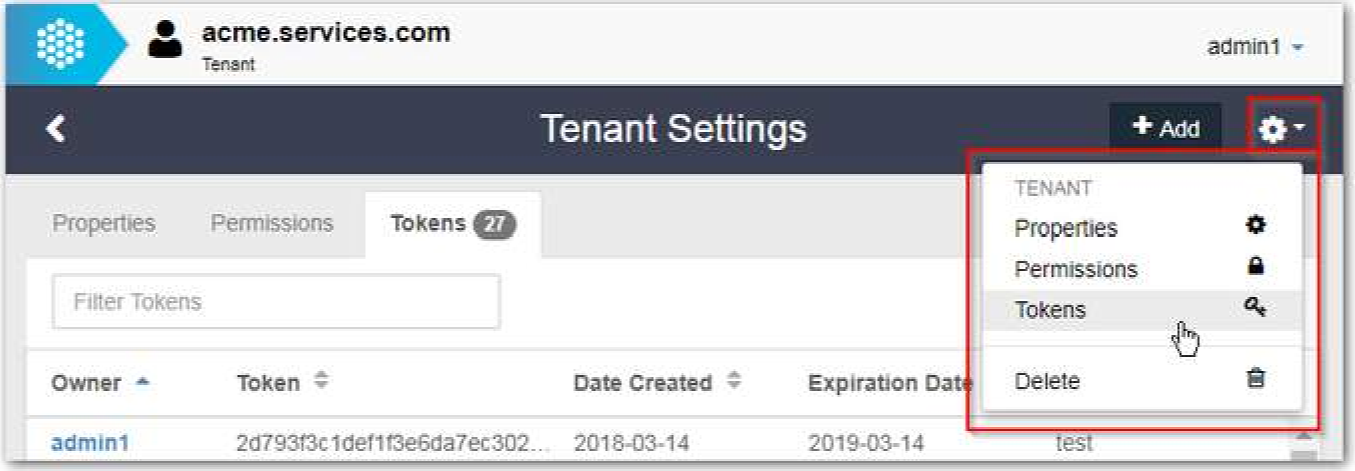
Usage summaries remain in view

Usage charts toggle in and out of view

Name	Owner	Storage Date	Size	Type
caringo-cs-docs-logo.png	admin1@	2016-07-11 9:25:21 PM	4.37 KB	image/png

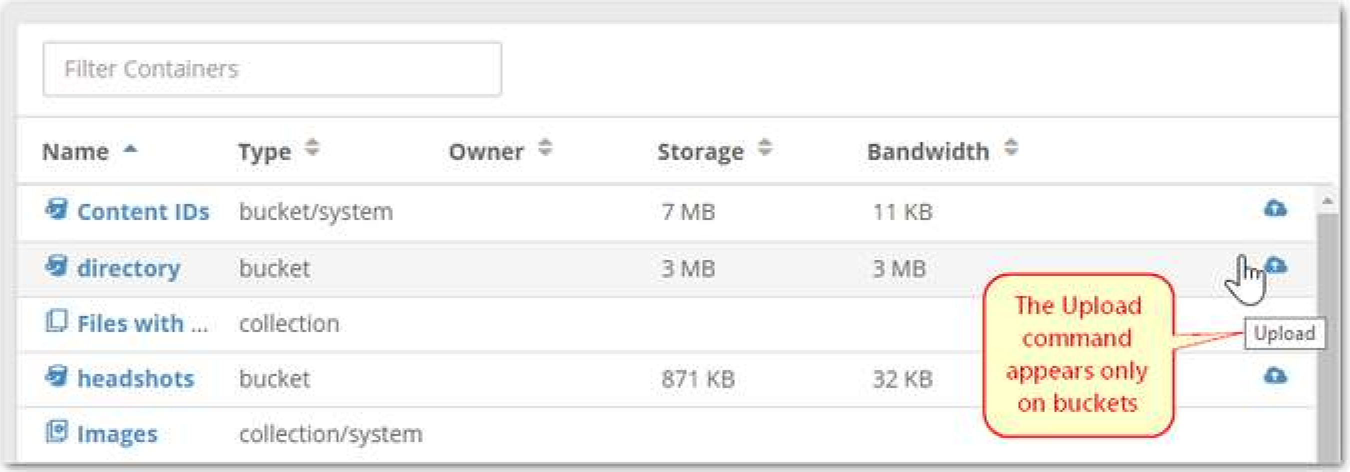


Commands and Properties — For consistency and simplicity, the commands and properties are all unified under the gear icon:



Owner	Token	Date Created	Expiration Date	
admin1	2d793f3c1def1f3e6da7ec302...	2018-03-14	2019-03-14	test

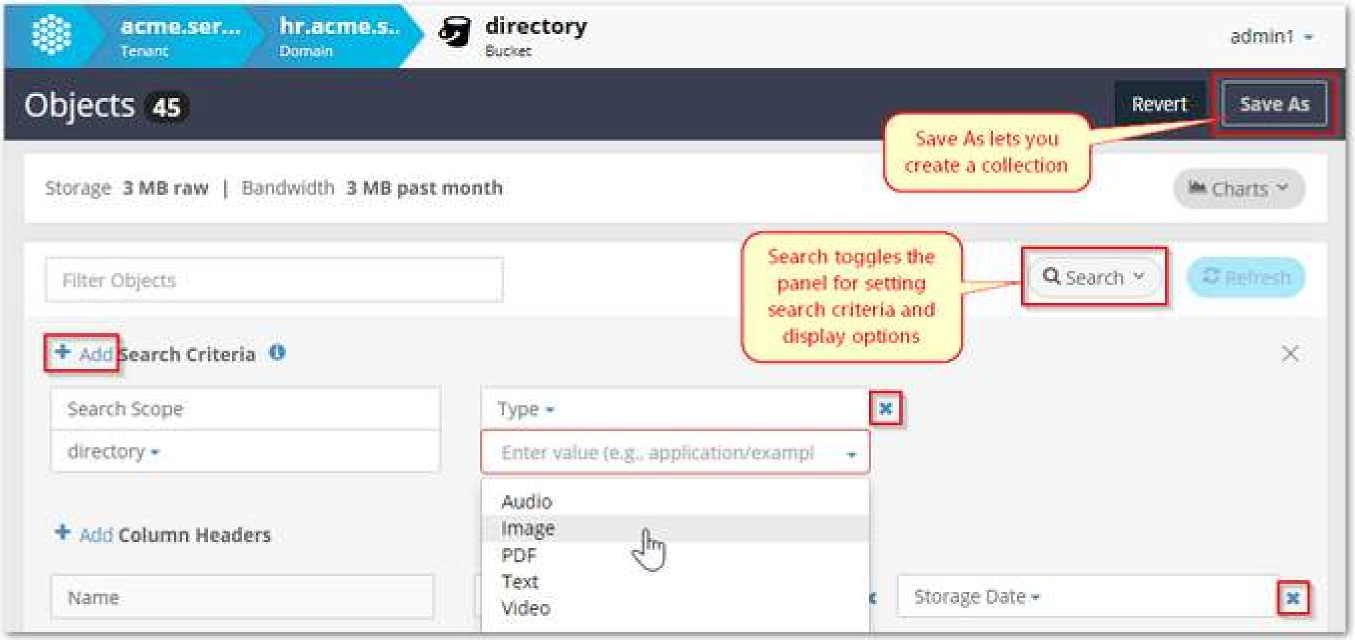
Commands such as Upload appear just where needed:



Name	Type	Owner	Storage	Bandwidth
Content IDs	bucket/system		7 MB	11 KB
directory	bucket		3 MB	3 MB
Files with ...	collection			
headshots	bucket		871 KB	32 KB
Images	collection/system			

The Upload command appears only on buckets

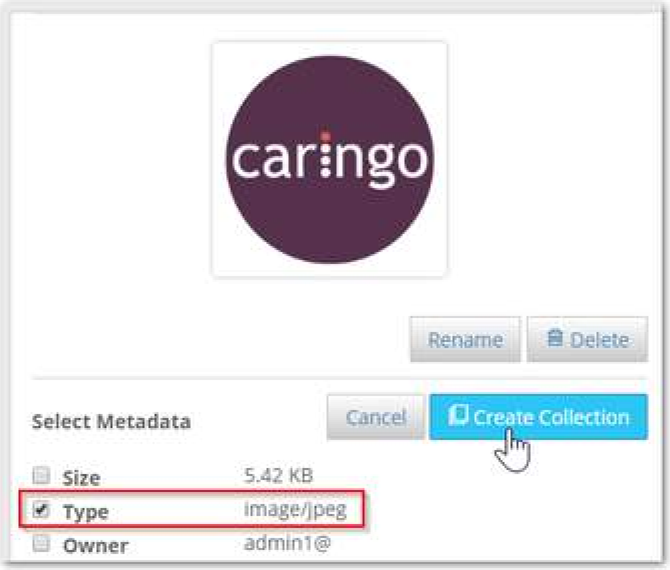
Search and Collections — Handling of search collection creation and filters has also improved, with the Filter and Search functions unified in a collapsible panel that you can expand by clicking Search:



Save As lets you create a collection

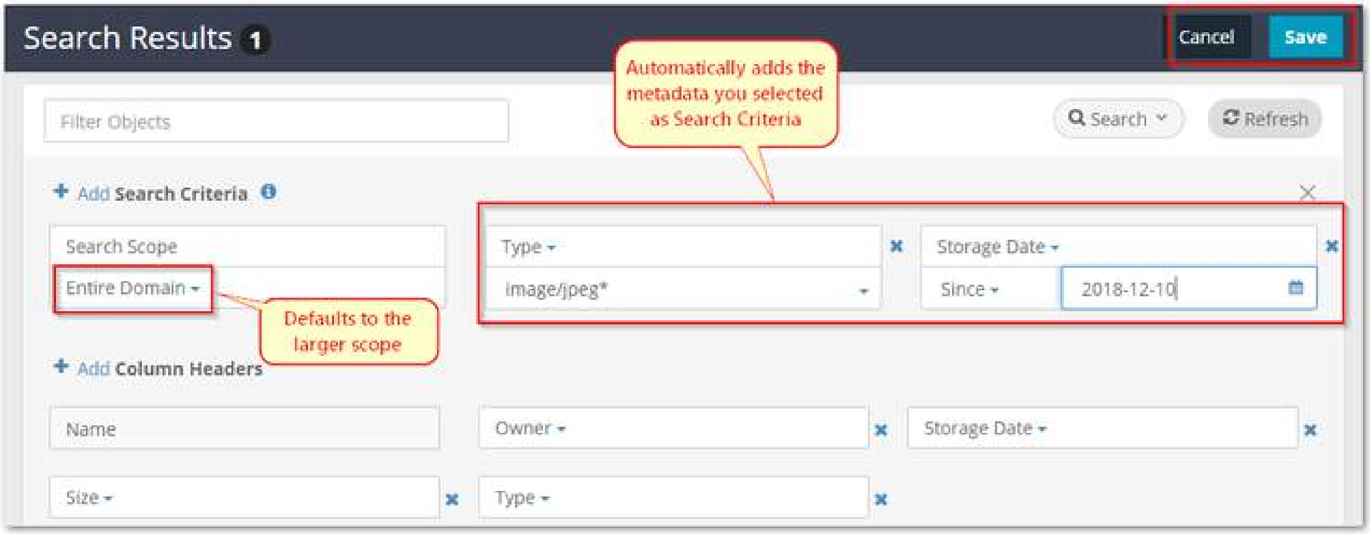
Search toggles the panel for setting search criteria and display options

By selecting Create Collection on the object view, you can create a metadata-based search in a single click, which greatly speeds up the design of your search criteria:



Select Metadata

☐ Size 5.42 KB
☑ Type image/jpeg
☐ Owner admin1@



Automatically adds the metadata you selected as Search Criteria

Defaults to the larger scope

Search Scope: Entire Domain

Type: image/jpeg*

Storage Date: Since 2018-12-10

Column Headers: Name, Owner, Storage Date, Size, Type

Known Issues

- After upgrading the Content UI (Portal) and/or Storage UI, the cache must be cleared to get the new version. Either shift-Reload the page or clear the browser cache, then verify that the About page shows the new version. A logout does not fix it nor is it necessary. (UIC-222)
- When uploading files from iOS devices, all of the filenames will be "image.jpg". You may upload these as UUIDs or upload one file at a time with a different filename prefix. (UIC-188)
- The login button for the Content UI may require a second click to proceed. (UIC-98)
- New tokens that have a user-supplied description will not initially show the description in the list of tokens. Workaround: Refresh the list of tokens after adding a new token. (UIC-43)
- To create a collection from metadata that includes non-ASCII characters, you must create it from the domain page. The ability to create a collection from non-ASCII metadata on the object details page is not currently supported. (UIC-31)

- For any collections that erroneously show the bucket icon, make a small edit and resave them, which causes them to be fully repaired. (UIC-24)

SwarmNFS Release Notes

Caringo SwarmNFS is a lightweight file protocol converter that brings the benefits of Swarm's scale-out object storage to NFSv4, seamlessly integrating files and object storage. With SwarmNFS, you can securely store and access data via NFSv4, S3, HDFS, and SCSP/HTTP, making it possible to organize billions of files coming from different protocols, distribute data to different locations, and search all of the files at once.

SwarmNFS brings the benefits of native object storage to NFS, but it is not a complete replacement for all traditional file (NAS/SAN) needs. In particular, note the following:

- Rapid updates — Frequent file updates, such as updates to databases, video editing, and storage of active vmdk files will not perform well and are not recommended.
- Large files — SwarmNFS performs well with files up to 30GB and best with files of 10GB and smaller; however, writing files of 100GB and greater is regularly tested. Reading from files (objects) of any size is fully supported.
- Versioning — Object versioning is supported (the last version written by any method becomes the current version), but it makes heavy demands on storage resources; enabling it in a SwarmNFS context is not recommended.

SwarmNFS 2.1

New Features and Changes

- To generate performance data, SwarmNFS now has profile logging, which is a configuration option that is disabled by default and hidden from the UI. Enable this logging only as directed by Caringo Support: once you have generated your logs, send them to Support, which has tools to analyze your read performance. (NFS-719)
- SwarmNFS has significantly improved the performance of sequential reads. (NFS-714)
- Logging for audit purposes has been improved. Open, delete, and rename operations generate NIV_EVENT-level messages in the standard SwarmNFS log. (NFS-684)
- When configuring SwarmNFS exports, you can now define default Owner, Group, and ACL to apply to any objects and synthetic folders that are created externally without preset POSIX permissions attached via metadata. (NFS-610)
- SwarmNFS now has a global hard/soft memory limit to work in conjunction with each export's own configured limits, to make better use of NFS server resources. Multiple exports on a single server now share the globally allotted buffer memory, rather than each carving out its own private buffer memory. (NFS-511)
- SwarmNFS supports the Linux `cp` command for copying metadata only (`cp file1:metadata file2:metadata`) and data only (`cp file1:data file2:data`), creating a new destination file with 0 bytes if needed. (NFS-469)

Known Issues

- Externally-written custom headers may not appear in :metadata reads. Workaround: To trigger ES to pick up an external update, also set the `X-Data-Modified-Time-Meta` header to the current time (in seconds since epoch). (NFS-692)
- Exports defined with different domains but the same bucket name do not behave as unique exports. (NFS-649)
- An invalid bucket name entered for an export in the UI will fail silently in SwarmNFS (config reads, export generates, client mounts, 0-byte writes and directory operations appear to succeed) but will fail on requests to Swarm Storage. (NFS-613)
- On startup, SwarmNFS may generate erroneous and harmless WARN level messages for configuration file parameters, such as config_errs_to_log :CONFIG :WARN :Config File (/etc/ganesha/ganesha.conf:17): Unknown parameter (Path) (NFS-289)
- SwarmNFS supports exclusive opens of a file (O_EXCL and O_CREATE) but does not support exclusive reopens (EXCLUSIVE4). (NFS-69)
- To prevent problems resulting from SwarmNFS disconnects or shutdowns, the Storage setting `health.parallelWriteTimeout` must be set to a non-zero value, such as 1209600 (2 weeks). (NFS-63)

SwarmNFS 2.0.2

- Fixed: Issues existed with directories that included spaces in their names. (NFS-593)

SwarmNFS 2.0.1

SwarmNFS 2.0.1 must be used with a Swarm cluster running Storage 9.5+ and with Storage UI 1.2.4.

New Features and Changes

- Performance is improved for how quickly external object updates appear in SwarmNFS listings.

Known Issues

- An invalid bucket name entered for an export in the UI will fail silently in SwarmNFS (config reads, export generates, client mounts, 0-byte writes and directory operations appear to succeed) but will fail on requests to Swarm Storage. (NFS-613)
- Cloud Security Authentication type Session Token is not yet available, although it appears as an option in the export definition.
- Reading metadata over NFS using `{filename}:metadata` is supported, but editing of object metadata over NFS is not yet supported.
- To prevent problems resulting from SwarmNFS disconnections or shutdowns, the Storage setting `health.parallelWriteTimeout` must be set to a non-zero value, such as 1,209,600 (2 weeks). (NFS-63)
    - Note that changing this setting affects S3, which defaults to keeping uncompleted multipart uploads indefinitely.
- To use SwarmNFS with Storage 9.5.0, set `scsp.keepAliveInterval = 45`. For best results, set Request timeout for each export to 90, so that it is at least twice the value of `scsp.keepAliveInterval`. (NFS-535, SWAR-7917)

SwarmNFS 2.0.0

SwarmNFS 2.0.0 must be used with a Swarm cluster running Storage 9.5+ and with Storage UI 1.2.3.

New Features and Changes

- Swarm Content Gateway is now supported. The SwarmNFS export configuration in Storage UI now supports Content Gateway in addition to Direct to Swarm. The Cloud Security section of each export configuration lets you set up the method that best fits your situation: Session Token (token admin credentials with expiration), Single User (user, password, and token), or Pass-through. See SwarmNFS Export Configuration.
- The defaults for NFS timeouts have been shorted to improve error handling. See SwarmNFS Export Configuration. (UIS-775)

Known Issues

- When creating an export in the UI, you need to increase default timeouts: in the Advanced Settings, set the Retries Timeout, Request Timeout, and Write Timeout all to 90 seconds.
- Cloud Security Authentication type Session Token is not yet available, although it appears as an option in the export definition.
- Reading metadata over NFS using `{filename}:metadata` is supported, but editing of object metadata over NFS is not yet supported.
- To prevent problems resulting from SwarmNFS disconnections or shutdowns, the Storage setting `health.parallelWriteTimeout` must be set to a non-zero value, such as 1,209,600 (2 weeks). (NFS-63)
- To use SwarmNFS with Storage 9.5.0, set `scsp.keepAliveInterval = 45`. For best results, set Request timeout for each export to 90, so that it is at least twice the value of `scsp.keepAliveInterval`. (NFS-535, SWAR-7917)
- Issues exist with feeds that were defined to use a non-default admin password. (UIS-759)
- Accessing unnamed objects is not supported.

SwarmNFS 1.2

SwarmNFS 1.2 must be used with a Swarm cluster running Storage 9.3.1+ and with Storage UI 1.2.1.

New Features and Changes

- Symbolic links (soft) are now supported.
- Demo clusters or those running on slower hardware or VMs are now supported.
  - Because slower hardware/VMs may require a longer update delay in order to operate correctly, the configuration now includes the setting `Scsp/UpdateDelay`.
  - See the Implementation Notes in SwarmNFS Server Installation.
- Fixed: Symbolic links to files did not return metadata if read using the ":metadata" suffix. (SNFS-346)
- Fixed: Generating core files can now be enabled and disabled via the nfs-ganesha.service file or through the system-wide configuration. (SNFS-297)

Known Issues

- When large writes are in progress, directory listings may at times appear to hang but will complete successfully.
- Accessing unnamed objects is not supported.

SDK Release Notes

The Swarm Software Development Kit (SDK) simplifies integration with Swarm by providing client library support for specific Simple Content Storage Protocol (SCSP) operations. The SDK assists developers by implementing a consistent set of features using a common API in each supported programming language.

- SDK version 9.1.0
- SDK version 6.1.5
- SDK version 6.1.4
- SDK version 6.1.3
- SDK version 6.1.2
- Limitations
- Deprecation Notices
- Application and Configuration Notes

SDK version 9.1.0

The SDK version was updated to reflect the version of Swarm testing and compatibility. This release includes the following enhancements and changes:

- The Java SDK is now built on Apache HttpComponents HttpClient 4.5.2 and HttpCore 4.4.4.
- The Java SDK testing was done against Java 8.
- The C++ SDK fixes a memory leak on redirect.
- The Python and C# SDKs now correctly handle a 202 response when the request is sent with `Expect:100-continue`. This is important for multipart completion handling with large numbers of parts (and therefore a large manifest in the request body). This fix means that all SDKs now properly support Swarm multipart completion and multipart copy-by-part requests.

SDK version 6.1.5

This release includes the following enhancements and changes:

- The Java SDK now includes several new classes and methods for facilitating remote replication for an object as well as synchronously writing to a local and remote cluster. Please reference SDK for Java for more details.
- The delete methods of `ScspDomain` and `ScspBucket` classes of the Java SDK now only add `?recursive=yes` if there is no recursive query argument on the call. This allows users to pass `recursive=now` to effect immediate content deletion for all content in a context.
- The `ScspDomain.create` call no longer passes policy-* headers to the _administrators bucket in the Java and C# SDKs.
- The C# SDK now supports chunked reads for a POST response and returns the trailer headers from the response in the ScspResponse object headers. Importantly this means that the C# SDK supports multi-part completion POST.
- Support for Content Router enumeration has been removed from the C# SDK as Content Router is deprecated.
- The Python SDK client now supports chunked reads on POST responses, including for multi-part completion POSTs.

SDK version 6.1.4

This release includes the following enhancements and changes:

- The Java SDK now builds using Maven and depends on HttpClient 4.2.5.
- Support for Content Router enumeration has been removed from the Java SDK as Content Router is deprecated.

SDK version 6.1.3

This release includes the following enhancements and changes:

- The C# SDK now correctly handles empty trailer headers on a chunked encoding response.
- The C# SDK now includes a ConnectionPool instance to allow connection sharing between requests. Important: When finished with an SCSPClient instance, applications must now explicitly call Close to ensure connections are not kept open.

SDK version 6.1.2

This release includes the following enhancements and changes:

- This release contains performance refactoring and optimizations for Expect/Continue handling in .Net for the C# client. The pattern closely matches Caringo reference implementations and has shown significant improvements in throughput, correctness and transaction rates in testing.

Limitations

These are the known issues and operational limitations that exist in this release of the Swarm SDK.

- Supported operating systems. Only English versions of operating systems are supported. Other versions or distributions, including languages other than English, are not currently supported.
- Swarm Locator Some languages, like Java, might include examples for how to use other locators like mDNS but these should only be considered examples and should be independently tested and verified.
- C++ Integrity Seal Hash Upgrades Due to an issue with the way curl handles long trailer headers, upgrading an integrity seal hash with the C++ client can fail occasionally.
- Using Range headers. Java, C++, and Python language implementations enable you to specify a Range header without bytes=, which is in violation of RFC 2616, section 3.12. The C# implementation does not have this issue. Code examples provided with each language show the correct way to specify a Range header. See the headers.AddRange example in the RunReadExamples method.

Deprecation Notices

This section lists functions that are deprecated and are subject to being removed in future SDK releases.

- The functions NoCastorNodesLocatorError.getFriendlyError and getFriendlyError in proxyLocator.py have been removed.
- Deprecated in SDK version 1.2: The uuid parameter has been replaced by path. Examples from SDK sample code follow:
- Java:
  - SDK 1.1: ScspResponse rcResponse = client.readMutable(uuid, "", outputStream, args, new ScspHeaders());
  - SDK 1.2 and later: ScspResponse rcResponse = client.readMutable("", uuid, outputStream, args, new

ScspHeaders());

- Python:
  - SDK 1.1: rcResponse = client.readMutable(uuid, fread, None, None)
  - SDK 1.2 and later: rcResponse = client.readMutable("", fread, None, None, path=uuid)
- C++:
  - SDK 1.1: client.readMutable(uuid, &outputStream, &response);
  - SDK 1.2 and later: client.readMutable("", &outputStream, &response, NULL, NULL, uuid);
- C#:
  - SDK 1.1: ScspResponse rcResponse = client.ReadMutable(uuid, "", outputStream, args, new ScspHeaders());
  - SDK 1.2 and later: ScspResponse rcResponse = client.ReadMutable("", uuid, outputStream, args, new ScspHeaders());

Application and Configuration Notes

Special attention should be paid to the following items when developing Swarm client implementations.

- ScspClient chunkSize parameters support in C++. ScspClient in all languages, including C++, supports the following parameters: getChunkSize, setChunkSize. However, unlike other languages, curl does not support explicitly setting how many bytes are sent at a time. curl provides a buffer and the buffer's length but does not enable the SDK to set the size of the buffer.
- C# Write, Update, Append. A Write, Update, or Append using the C# SDK client that encounters an error response, an ScspWebException might be thrown. This can occur with a 400 response from the cluster, or on any error response (code 400 and greater) when using the SCSP Proxy. This behavior is caused by the way that .Net internally handles a connection closing while writing data to a peer.
- Java recompile required. Because of internal changes to the Java SDK client, you must recompile your Java code against the classes provided with the SDK.
- C# connection timeout. Increasing the connection timeout might help alleviate write failures on large objects due to too many retries of cancelled requests. For more information, see the chapter on C# in the SDK Overview.
- Pre-emptive authorization (from RFC2617). Pre-emptive authorization enables client applications to generate an authorization header initially, bypassing the server's authentication challenge. Language-specific implementation details for the Swarm SDK follow:
  - Python and C++. Pre-emptive authentication works.
  - Java. Pre-emptive authorization does not work. Every request for a protected resource generates an initial 401 (Unauthorized) response from Swarm.
  - C#. Pre-emptive authentication fails with requests that must be authenticated in different domains.
- For best results, build curl using Visual Studio C++ 2008 or earlier. You might have problems building the latest curl version with Visual Studio C++ 2010. For more information, see the curl install page.
- Failed Integrity Seal Validation Swarm will close the connection if an integrity seal fails validation on a read, which is shown in the client as an I/O error.
- C++ character encoding. When you pass in a URI path as a string using the C++ SDK, you must use the string class. If the path needs includes non-ASCII characters, these characters must be UTF-8 encoded by the caller.
- Java, C++, C#, and Python character encoding. When you pass in a URI path using the SDK, you must escape a backslash character (\) with %5c.
- Java ResettableFileInputStream In the Java client, FileInputStream() cannot be used; instead, use

ResettableFileInputStream, which is located in CAStorSDK-src\com \caringo\client.

Swarm Deployment

Caringo Swarm

Swarm combines the scalable software-defined object storage of Swarm Storage with components that support many types of implementations.

To implement Swarm, you install its components in this order:

Platform Server	Node for site-wide management and services	Install (CSN Install, Upgrade)
Storage Cluster	Cluster for Swarm storage nodes	Requirements, Network, Install, Configure
Elasticsearch	Cluster for search and historical metrics	Requirements, Prepare, Install, Configure; Metrics Install
Content Gateway	Gateway for cloud-based client access (S3)	Requirements, Install, Configure
Storage UI	Website for storage cluster management	Install (CSN Scenarios)
Content UI	Website for cloud content management	Install
SwarmNFS	Optional connector for NFS clients	Install, Configure

Before installing any Caringo packages, be sure to complete the planning and preparation of your Swarm environment.

- Migrating from Traditional Storage
- Use Cases and Architectures
- Deployment Planning
- Deployment Process
- Network Infrastructure
- Hardware Setup
- Platform Implementation
- Storage Implementation
- Elasticsearch Implementation
- Swarm Storage UI Installation
- Content Gateway Implementation
- Content UI Installation
- SwarmNFS
- Caringo Drive

Migrating from Traditional Storage

Advantages of Object Storage

There are features of an object storage system that make aspects of traditional file systems obsolete.

- It has a single large pool of storage that does not need to be backed up (in fact, it is often too large to be).
- It offers enhanced metadata characteristics, which can be extended to suit the user's needs.
- It includes large-scale, high-performance searching based on that metadata.

Never-ending storage systems

Traditional file systems and storage systems have hard limits at some point. Whether it is at the volume / block layer level or at the partition level, there will always be an upper limit in how large you can make a LUN or where a partition begins to become unmanageable due to size. Object storage offers a seemingly limitless namespace + storage layer to place data.

As filesystems are developed, they push the limits further and further. We see the max file size and max volume size grow regularly. These are common file systems and their limitations:

Filesystem (released)	Max Filename	Max File Size	Max Volume Size
EXT (1992)	255 bytes	2 GiB	2 GiB
EXT2 (1993)	255 bytes	16 GiB to 2 TiB	2 TiB to 32 TiB
EXT3 (2001)	255 bytes	16 GiB to 2 TiB	2 TiB to 32 TiB
EXT4 (2008)	255 bytes	16 GiB to 16 TiB	1 EiB
XFS (1994 - SGI, 2001 linux kernel, 2014 widespread adoption)	255 bytes	8 EiB	8 EiB

FAT16 (1984)	255 UCS-2 characters	32 MiB	16 MiB (FAT16 x 2 GiB)
FAT32 (1996)	255 UCS-2 characters	2 GiB	512 MiB (FAT32 x 8 TiB)
NTFS (1993)	255 characters	16 EiB	16 EiB
ReFS (2012)	255 UTF-16	16 EiB	3.76 ZiB

You can see over time and based on differing technologies how the size of an individual volume has grown. If ReFS + Xfs can grow to thousands of petabytes, why the need for object storage?

Although theoretically NTFS can reach 16 EiB, the maximum LUN size that can be addressed by Windows Server 2012 was 64 TB due to chkdsk and snapshotting limitations, if you wanted to back it up in a timely fashion.

https://support.microsoft.com/en-us/help/2967756/usability-limit-for-volume-shadow-copy-service-vss-in-windows

Likewise, ext4 in older Linux kernels had similar limits. While in theory volumes larger than 16 TB could be created, the tools that were used to operate and fix issues with them initially could not work with these larger volumes. Granted, many of these issues have been fixed in newer kernels and with tool updates, so this is no longer a problem.

http://blog.ronnyegner-consulting.de/2011/08/18/ext4-and-the-16-tb-limit-now-solved/

Typically, large data LUNs can only be created by aggregating multiple disks together via hardware or software RAID technologies and accessing them via a fast interconnect, like fiber or iSCSI. These RAID volumes will have their own limits and durability characteristics, which makes LUN sizing difficult. Different SAN manufacturers will have different limits on the ideal sizes and distribution of LUNs. Often, IT administrators will need to prioritize the type of data protection level and speed when deciding to commission new storage. It's rarely as simple as making the largest volume possible and offering it out to users to carve up in their own way.

In contrast, Caringo's clusters are one large volume of storage that can share the same protection profile but also can use different protection profiles within the same cluster of storage.

Bullet-proof protection

When we talk about data protection in traditional terms, usually we’re talking about 1 or 2 disk failures for a single RAID volume in a SAN or local RAID group. Typically you can replace those failed drives and suffer decreased performance while the parity is rebuilt after a period of hours or days, depending on the size of the volume and the amount of data on it. Multi-disk failures are fairly common, and hard drives are getting larger and larger, leading to longer rebuild times and larger datasets and backup times.

Caringo’s object storage system is designed to sustain multiple disk failures and, depending on the design, multiple chassis / server failures. This is important when thinking about data protection as a whole.

Rich metadata

In today's storage landscape, it is increasingly the case that the information about the data is as important as the data itself, for analytics and other purposes. With a Caringo object storage solution, we can store up to 32 KB of custom metadata with each object. Compare this to a traditional file system, like NTFS or ext4, where the metadata for the file is fixed by the filesystem and generally only used to define system-side information (access times, owner, attributes, etc.).

That's not to say that extended metadata isn’t used outside of object storage. All sorts of custom file formats have been created to allow files to carry their own information like a passport. ID3 with mp3 and mp4 files, for example, and Raw image formats contain various metadata headers embedded in the filetype. Usually when the file contains the metadata itself, the application used to view the file will restrict what metadata is visible.

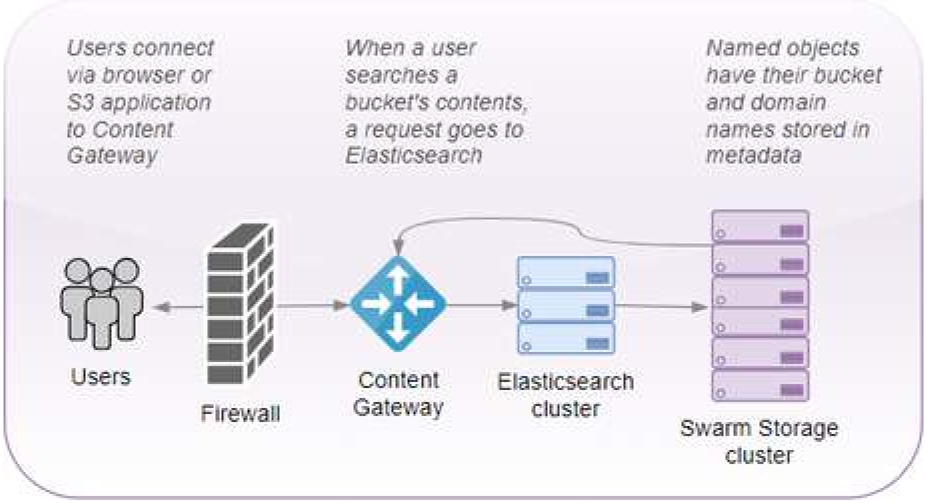
In a Swarm cluster, all metadata associated with a file is stored as header information on the file itself. The header info can be viewed via a simple HTTP HEAD of the file, requiring no special drivers or applications to do so.

Advantages of Deploying Content Gateway

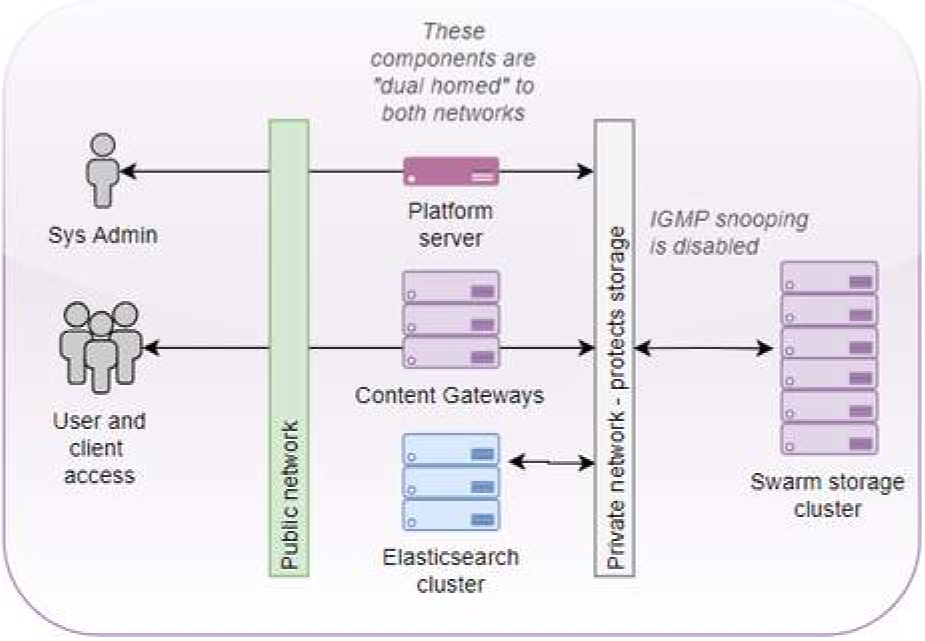
Implementing Swarm with Content Gateway provides the organization with authentication, a browser UI for end users, S3 protocol access, and enhanced multi-tenancy. Multi-tenancy (discussed below) can be a critical tool for dividing and delegating content access and structure within large organizations.

Below is a basic Swarm deployment that leverages Content Gateway:

- A 6-chassis Swarm cluster, for hardware resilience
- Elasticsearch cluster for dynamic searching
- Content Gateway

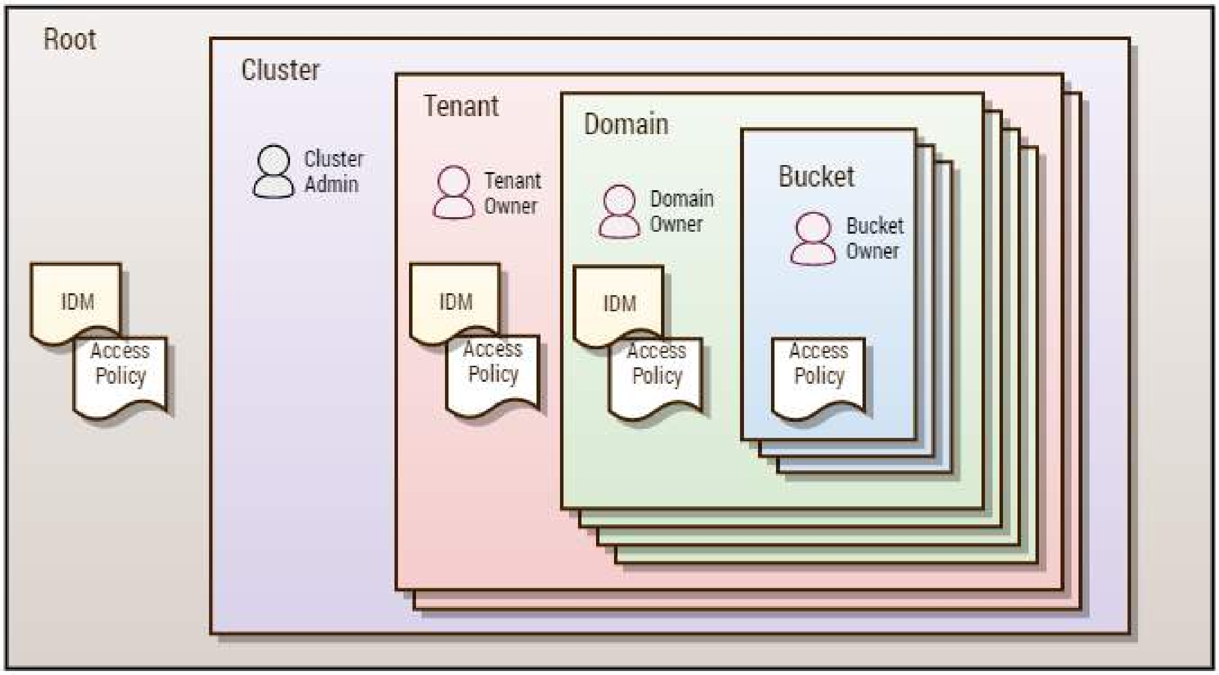

Users connect via browser or S3 application to Content Gateway
When a user searches a bucket's contents, a request goes to Elasticsearch
Named objects have their bucket and domain names stored in metadata
Users
Firewall
Content Gateway
Elasticsearch cluster
Swarm Storage cluster

The Swarm Storage cluster is protected within a dedicated private network, and all client and application traffic passes through Content Gateway:



Tenants, Domains, and Buckets

Swarm offers multiple levels of access, but let's focus on tenants, domains, and buckets:



- Tenant — A tenant is a hierarchy that owns one or more storage domains. Each tenant scope can define their own identity management system so that the users and groups within them are separated from those in other tenants. The tenant administrators have the ability to create and access storage domains on behalf of the tenant, and they can delegate management duties for the storage domains that they create. The tenant scope

does not store end-user data; it is only a meta store for information about the tenant, its users, and its storage domains.
- Domain — The domain scope is directly tied to a Swarm storage domain and is where end-user data is kept. The SCSP and S3 storage protocols create and use data within the domain scope. While the domain scope can inherit user and group identity information from its tenant, it also has the ability to define its own identity management system. The domain administrators can create and access all content within the storage domain. They can optionally delegate control of storage buckets to individual users or groups.
- Bucket — The bucket scope is directly tied to a bucket that exists within the Swarm storage domain. While access control policies can be defined for every bucket, there is no option for an identity management system definition at the bucket scope. All buckets with a domain share the domain's identity management system definition.

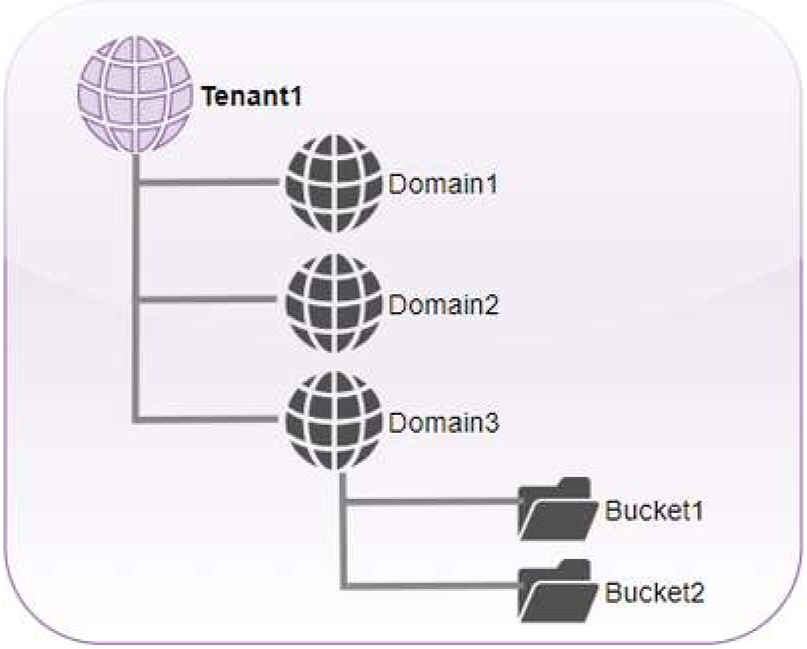
In short: A tenant holds multiple domains, and a domain holds multiple buckets.

Organizing by Tenant

Outside of multi-tenancy environments, tenants are useful for grouping similar storage areas in a cluster.

Single tenant, wildcard DNS

Here is a simple top-level structure:



Tenant1's auth and protection levels are inherited by the domains lower down. In this example, only Domain3 has buckets (represented here by folders).

> Note
> Even though the tenant is a special type of domain specific to Gateway, it is still a domain.

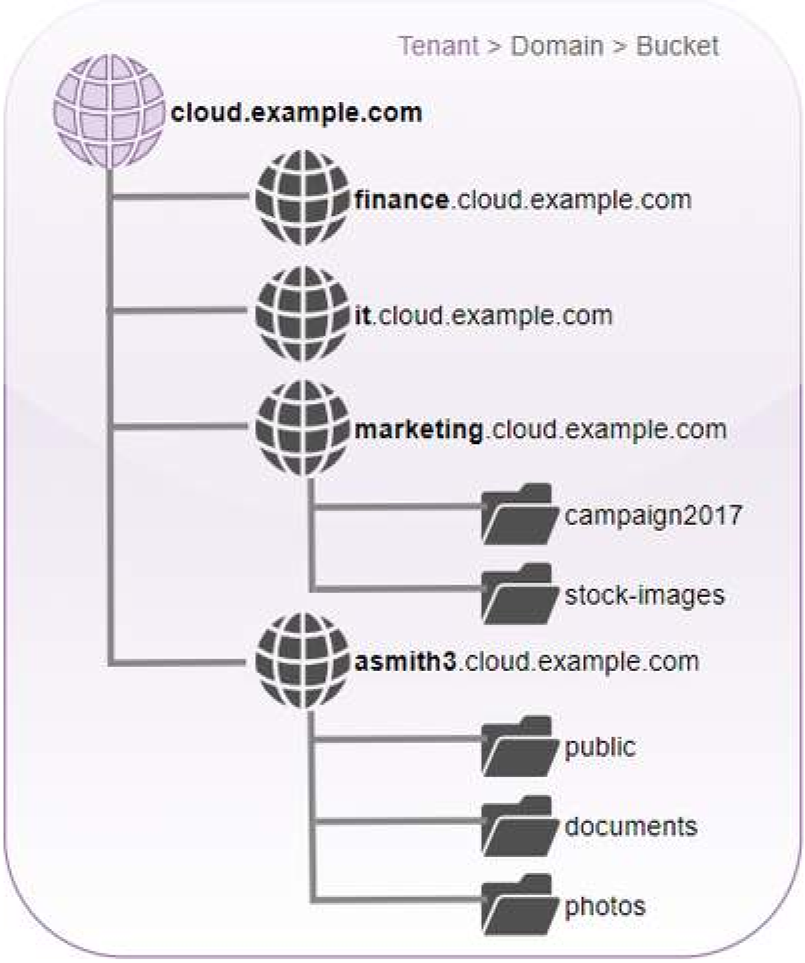
Each of these domains can also be fully qualified within the corporate DNS structure.

Tip

Use wildcards so that you don’t need to add DNS records for every new domain as they are created. This enables you to allow users to create domains of their own, and DNS resolution will happen automatically as long as the domains are created with a similar naming structure.

One domain per department, employee

In this example, we create a wildcard DNS record for the gateway's address: `*.cloud.example.com`

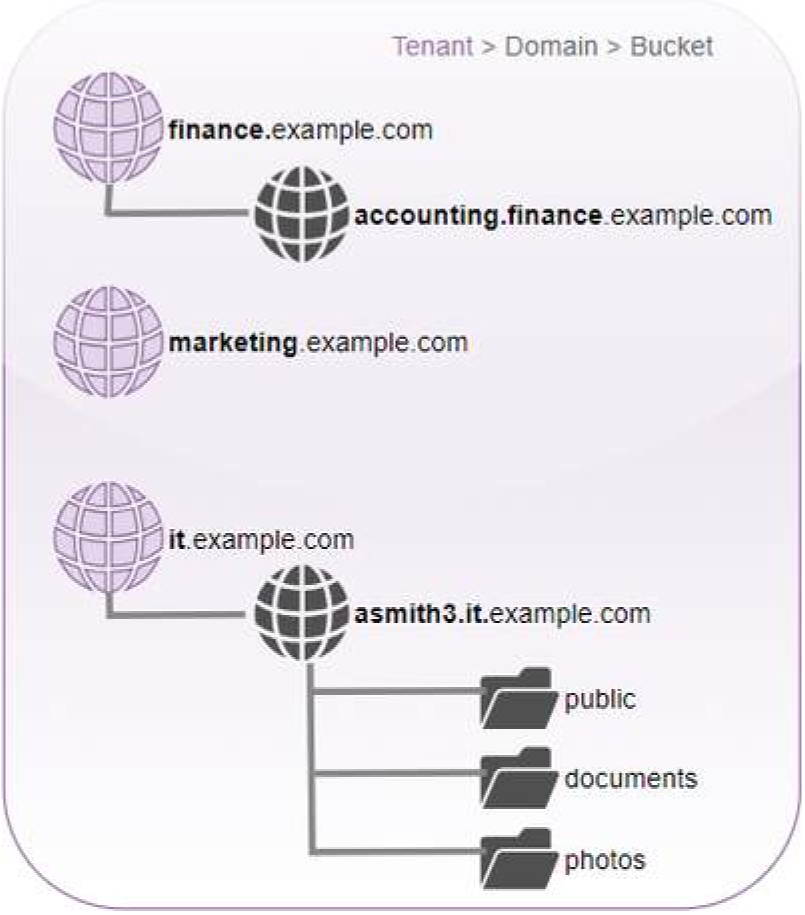


Each domain created here represents a single department in the organization. However, as there is no limit to the number of domains you can create within the storage cluster, you could also create a domain for every employee. In this example, the last domain is an employee domain: `asmith3.cloud.example.com`

Employees can create as many buckets as they wish within their own domains to further subdivide their content.

One tenant per division

For your organization, it may make sense to have more than one top-level tenant. You might give each corporate division its own tenant so it can create and control its own departmental and employee domains. That provides an additional level of organization and authorization to work with.



Ultimately, what it comes down to is how to create the most readable and the shortest path to the information that the users care about.

The division should make sense if read from a browser. The following URL would be easy to interpret and simple to get to:

```
http://accounting.finance.example.com/fiscalresults2017/data.xls
http://<dept>.<division>.<org>.com/<bucket>/<filename>
```

Migration Planning

Adapting the Legacy Structure

When migrating data from a traditional block storage or file-sharing solution, it is important to evaluate the structure that is already there and decide what makes sense to keep of that structure moving forward.

Users don’t like change. When a file server goes into an organization, there tends to be tribal knowledge about what goes where that gets ingrained into an enterprise. New users are given access to the “P” drive or the “docs” folder, and, bit by bit, they learn where things are and where to put things. So when it comes to implementing a new structure or talking about a new file server, if you were to ask a user how they would like the new system to look, they will usually say “Just like the one we have now!” That is a real challenge, and there's no easy answer to combat this.

But when doing a migration of any sort, know that this is the time to start to implement change. It's important to evaluate the old structure and see what's working and where there’s duplication, so you can eliminate it.

Best Practices for Restructuring

Don’t bulk move folders to pseudo folders

- An object store offers immense flexibility; bulk moving folders of files removes that flexibility and keeps the older structures. This makes it harder to change going forward.
- Any 1-to-1 movement will need to use pseudo folders, which are just prefixes to an object name. Pseudo folders make the objects harder to search for and result in difficult searches.
- Permissions and user attributes are on the object, not the folder. If a user creates a pseudo folder thinking they can share that and all files in it, they will be disappointed.

Convert pathnames using domains and buckets

- If you have a very long path name such as /year/month/day/filename, think about how that would best look in an object context. The shortest path would be to have the domain as the year, with the bucket being a month+day context. For example: `2017-hq-videos.example.com/Sep-13/videofile.mp4`
- If the date is in the filename, there is no need to have a date on a bucket name.

Use domains for data groups

- If you have a large amount of similar data or data that is always used in the same workflow, give it its own domain.

Use tenants/domains for applications

- If your organization uses a particular application whose dedicated data will only be used via that application, give the application its own tenant or domain.

Optimize for searches

- Collections are saved searches that, like buckets and domains, are unlimited to create. The scope of a search can be the entire domain or a specific bucket.
- When creating domains and buckets, avoid creating a structure that is too granular for large searches. For example, if you’re creating a bucket per day in a domain for a certain type of data, creating a bucket per hour might be excessive unless there's a lot in each bucket.

Planning Areas

Any migration project requires consultation with Caringo and planning around these key areas, such that all integration points in your environment can be listed and diagrammed:

Namespace	Strategy for mapping file systems to objects (discussed above) FQDN and DNS setup for Gateway (see Content Gateway Implementation)

Networking	Work out down to each port (see Setting up the Swarm Network) how all Swarm components will integrate, to uncover any design issues List application requirements and ensure applications can access storage regardless of network segment HTTP or HTTPS? Front-end load balancing or round robin?
Authentication	Will there be LDAP or Active Directory integration? How will the current ACL structure map to Gateway ACLs? (see Content Gateway Authentication an d Setting Tokens)
Swarm clients (optional)	SwarmNFS • Check minimum requirements if deployed client-side • Networking implications (Elasticsearch access and IP whitelisting) Caringo Drive (check minimum requirements) FileFly

Use Cases and Architectures

Most use cases for Swarm involve ingesting petabytes of unstructured data, such as image, video, and document files, which must be secured, preserved, searched, and retrieved on demand.

- Active Archive — video evidence, medical imaging, cultural media
- Cloud — cloud services and hosting (multi-tenant), backup to the cloud
- Content Delivery — social media (millions of photos per day), streaming video (millions of videos), content publishing (millions of images)
- Big Data — evidence analysis, medical insurance records and analysis, IoT/M2M and analytics
- Compliance — legal documents, court materials, digital evidence

Swarm supports many usage scenarios based on four fundamental access methods:

Direct Access	Native (SCSP API)	• Native client/application integration using a vendor API (RESTful HTTP 1.1 compliant) • Expectation that the application will work directly with the object store
Web Access	Content Gateway (S3 API)	• Data in the object store is presented via web browser (Content UI) • S3 endpoint is provided • Support for actions such as upload, download, and browse • Back-end access to the object store are native API calls

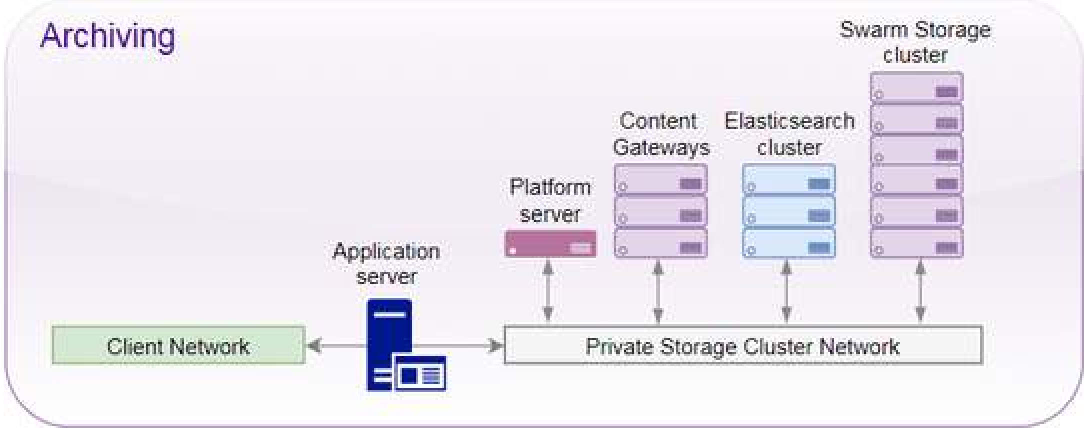
File Protocol Gateway	SwarmNFS (to Native SCSP)	- Provides translation of traditional file protocols (such as NFS and SMB) to object storage protocols - Usually translates to object store native API - Used as a “drop box” target for clients/applications coded to work with traditional filers - Advanced protocol gateway support for manipulation of metadata, in addition to data via traditional utilities (such as shell sessions) - Placement into object store supports alternate access methods (SCSP or S3) and metadata queries, listings, and collections
Automated Tiering	FileFly Swarm Hybrid Cloud Caringo Drive	- Application/agent integration (native API integration) - Agents live on data sources (such as filers) - Relationship between local file reference and data stored at object tier is maintained by agent software - Data is moved from local to object tier based on policy (scheduled or ad hoc) - Retrieval of data when client requests access is automatic and transparent

These are common architectures for object storage:

- Archiving
- Data tiering
- Remote replication and disaster recovery
- Managed service (“Storage as a Service”)
- Hybrid Cloud (local storage with Cloud)

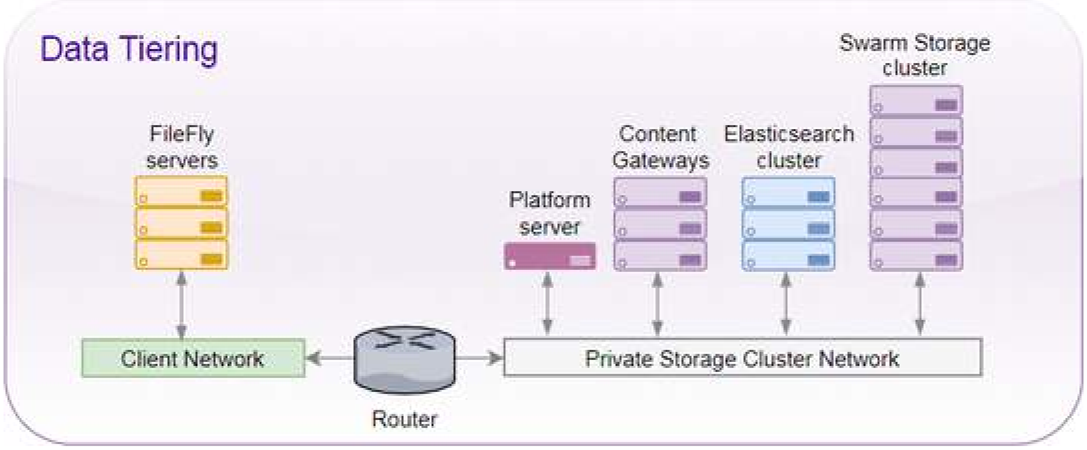
Archiving

- Medium- to long-term storage
- “Write once, read rarely”
- Library of unstructured data (documents, graphics, pictures, videos)
- Query and list, based on metadata tags
- Conduct “Data Lake” analysis (by pooling a vast amount of raw data in its native format)

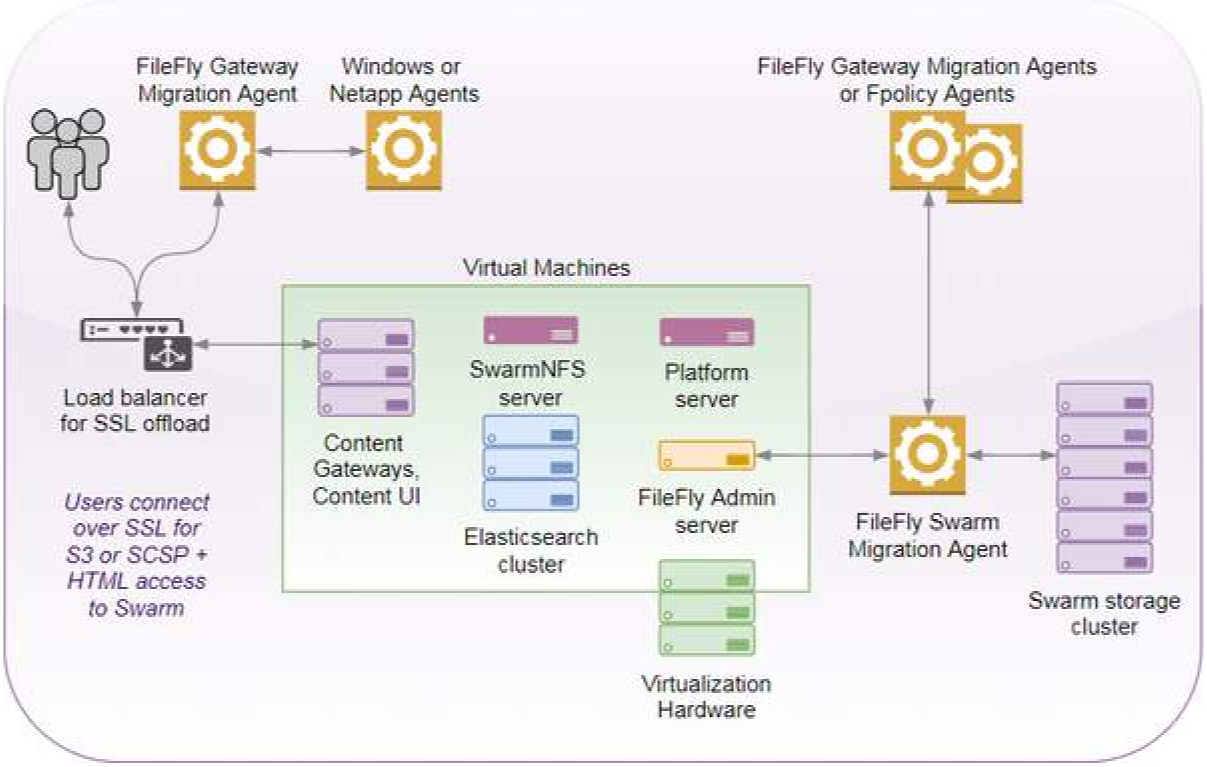


Data tiering

- Relocation of data from traditional filers to object storage
- Scheduled tiering based on policy
- Automated recall when access request is made
- Transparent access to the end user
- “Cheap and deep” object store tier to reduce filer expansion costs

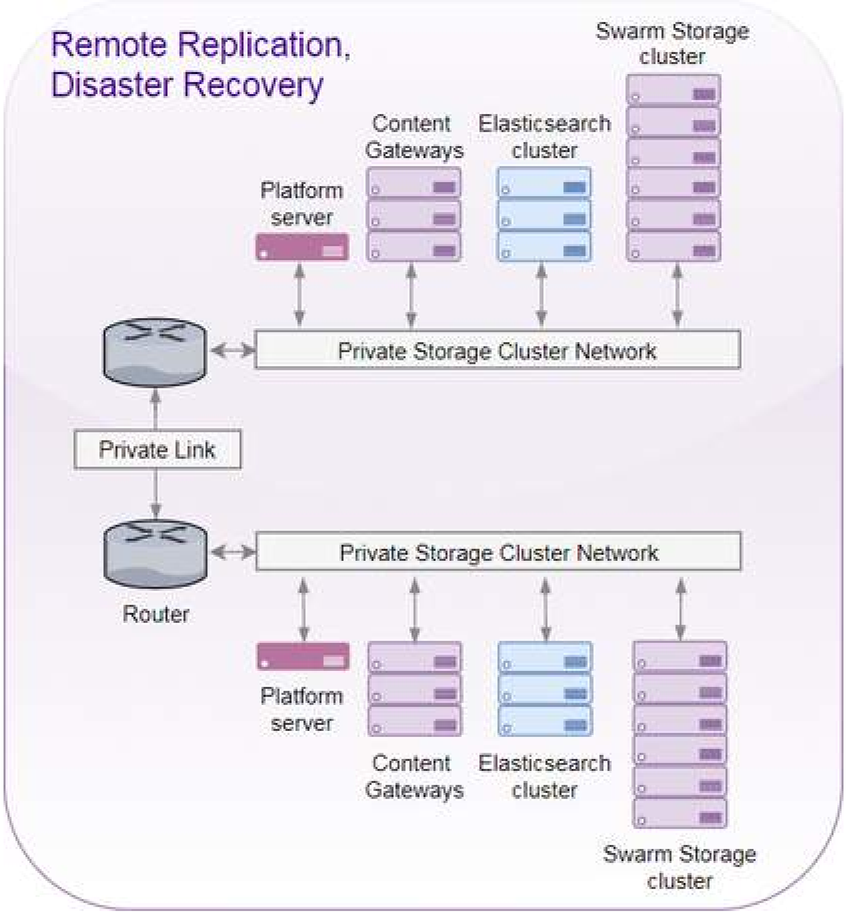


FileFly and Virtualization

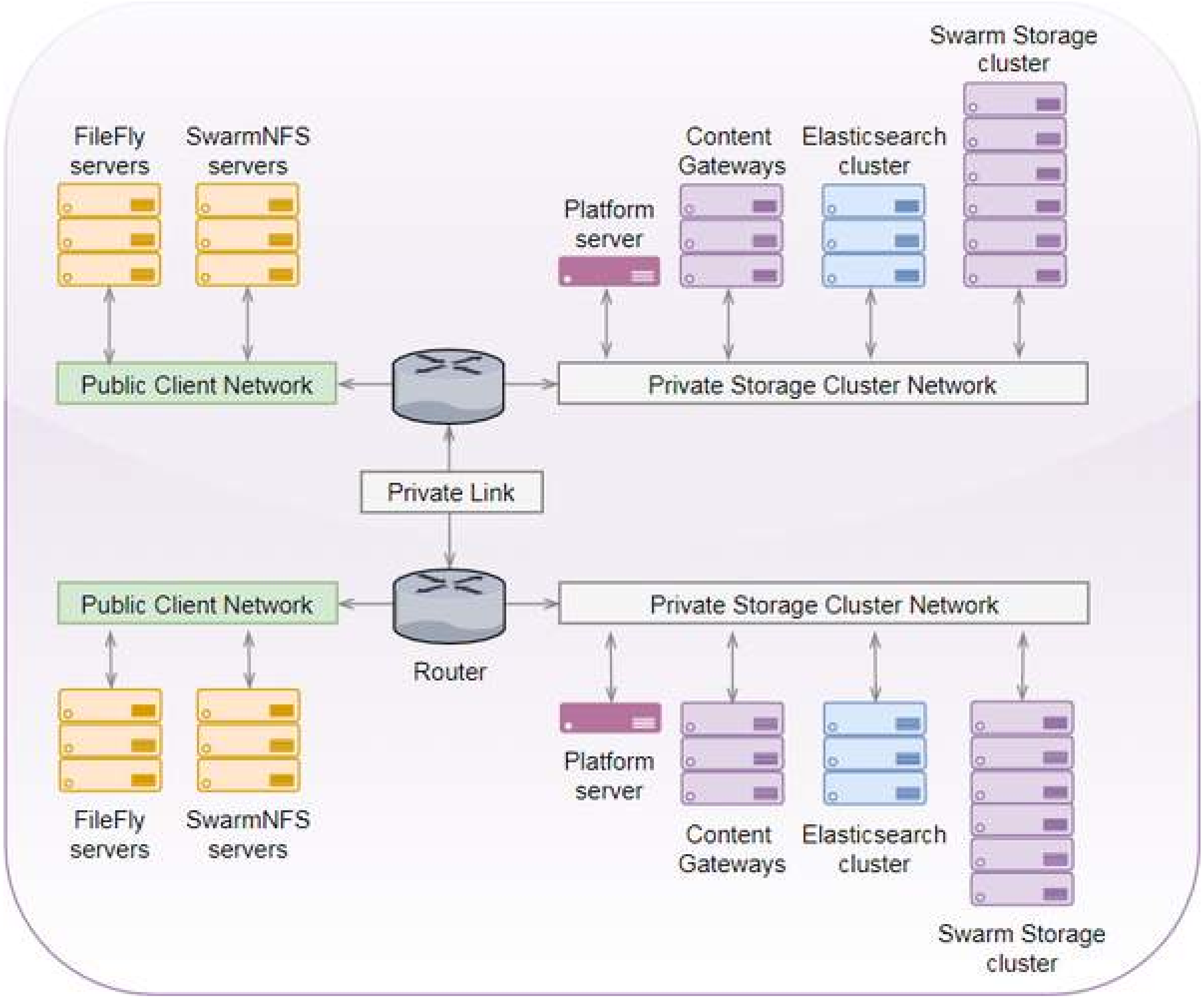

FileFly Gateway Migration Agent
Windows or Netapp Agents
FileFly Gateway Migration Agents or Fpolicy Agents
Virtual Machines
Load balancer for SSL offload
Content Gateways, Content UI
SwarmNFS server
Elasticsearch cluster
Platform server
FileFly Admin server
Virtualization Hardware
FileFly Swarm Migration Agent
Swarm storage cluster
Users connect over SSL for S3 or SCSP + HTML access to Swarm

Remote replication and disaster recovery

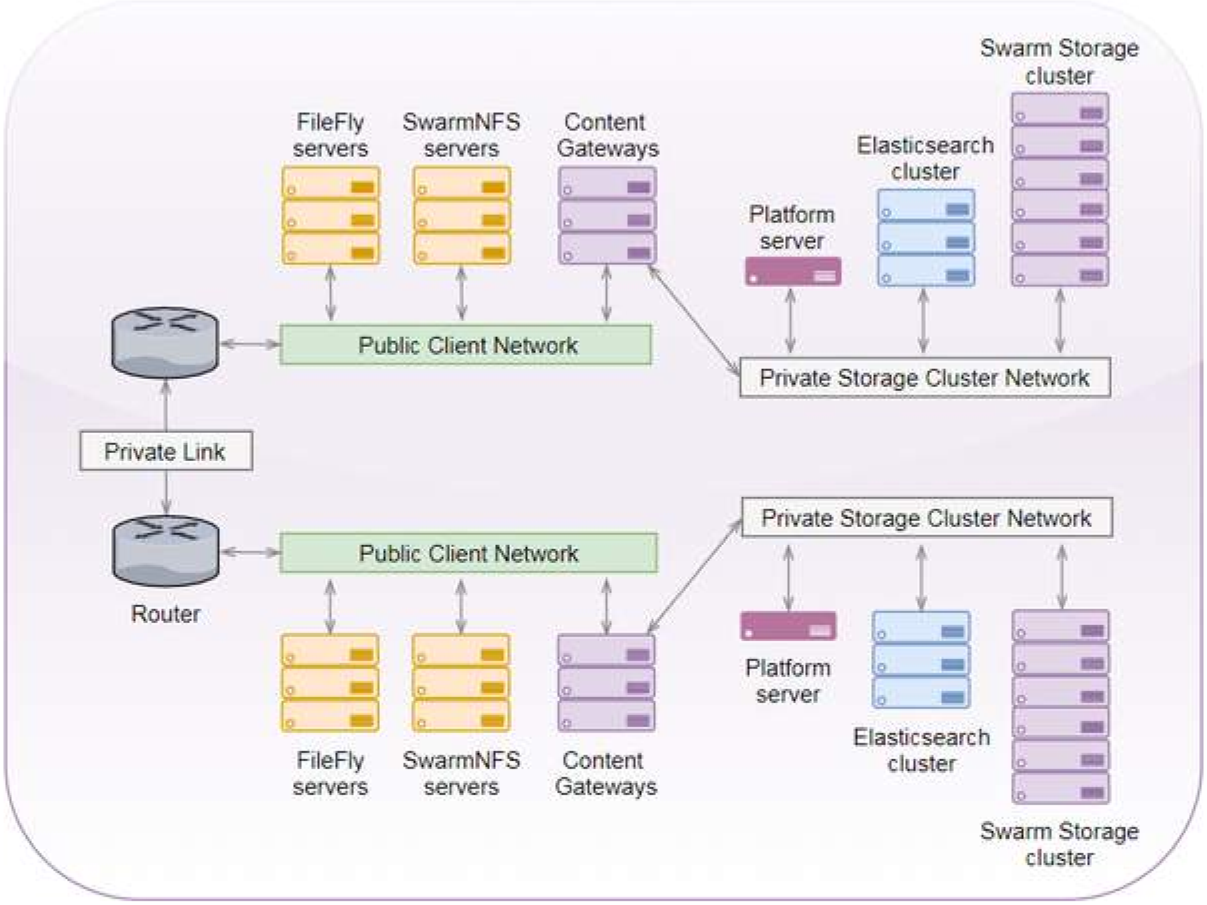
- Automated replication from a local object store to a remote object store
- Data is usually populated in a local store, then replicated to remote/DR
- “Hot” sites can also act as replication targets/DR for each other
- Can be whole site replication or policy based (per domain)
- Simple to complex replication topologies supported (site-to-site, M to N, single or bi-directional)



Dual Site with Single Interface

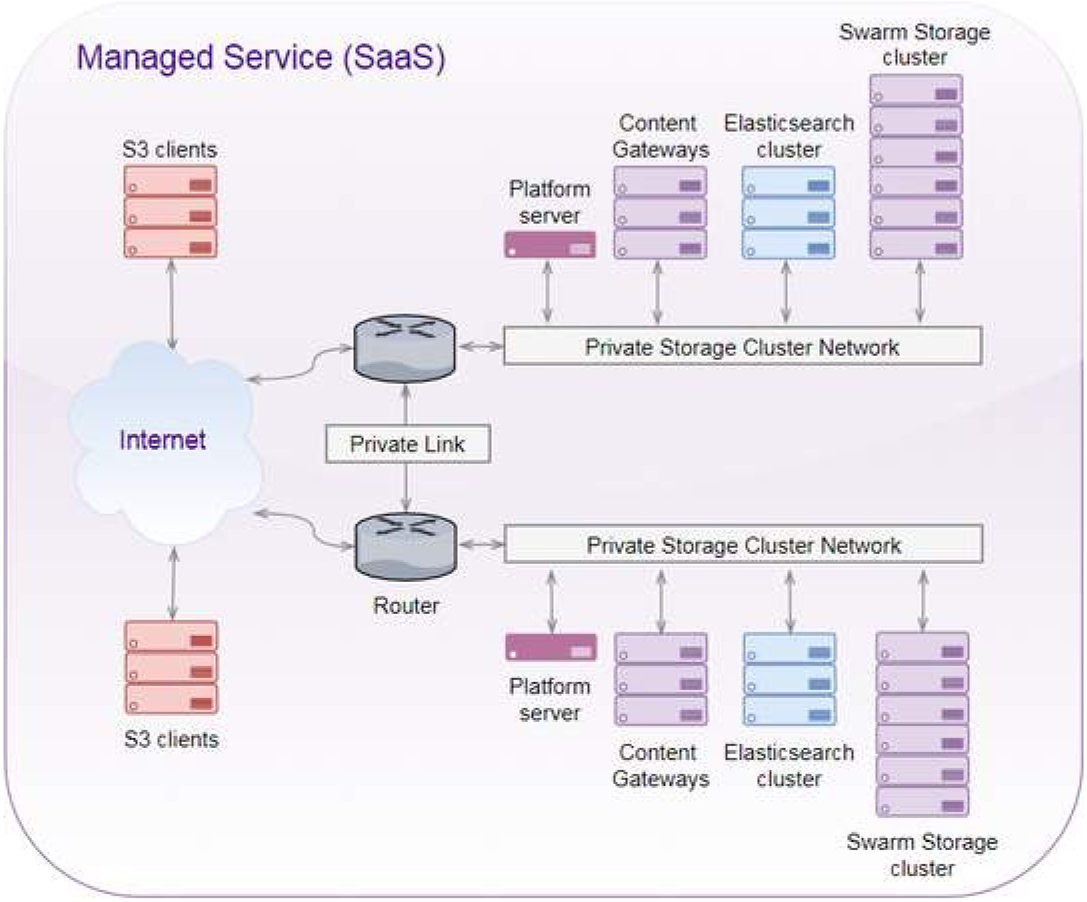


Dual Site with Dual Interface



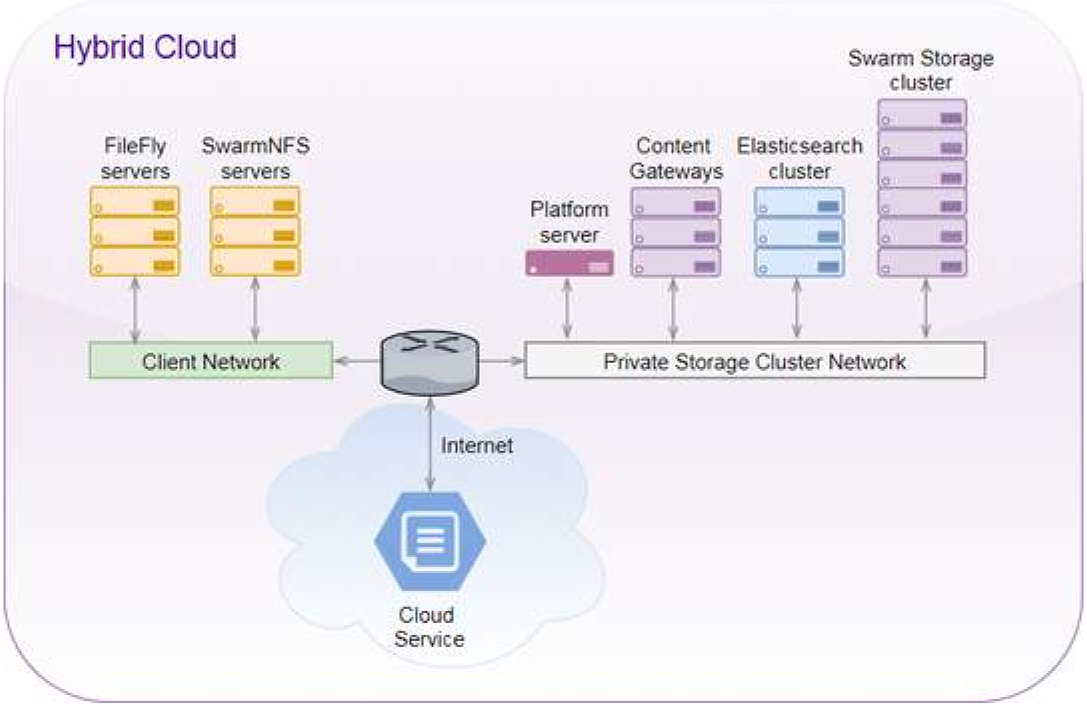
Managed service ("Storage as a Service")

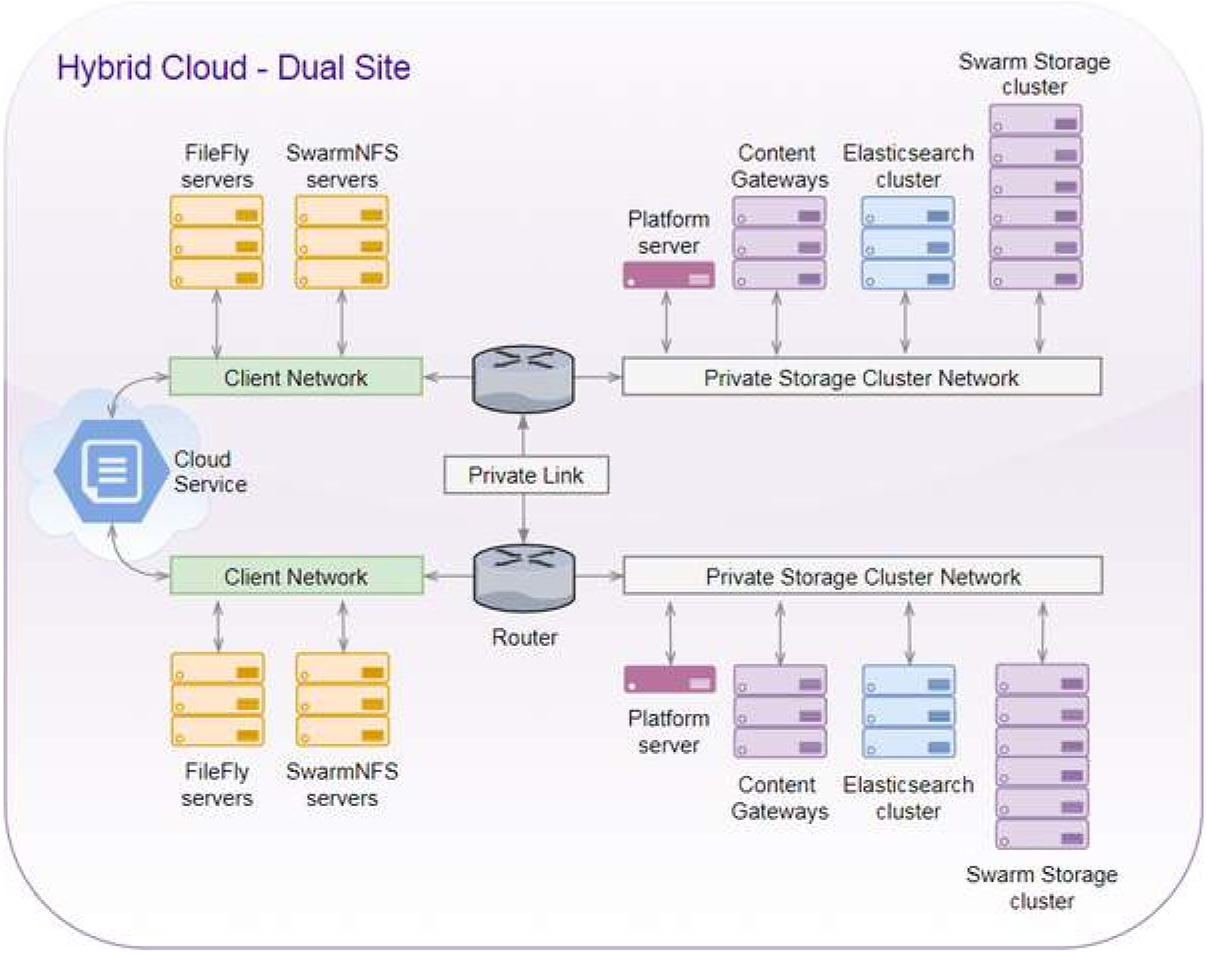
- Storage protocol endpoints made available to service subscribers
- Support for multiple RESTful access protocols
- SSL/TLS
- Provides authentication and authorization
- Allows for metering and billing
- Supports quota control
- Multi-tenancy (individuals, business organizations, business units)



Hybrid Cloud (local storage with Cloud)

- Local object store integrated with a cloud service endpoint (such as Azure)
- “Copy to Cloud” for backup and/or publication of data
- “Retrieve from Cloud” for data recovery
- Lower CapEx when meeting backup/replication/DR requirements


Hybrid Cloud
FileFly servers
SwarmNFS servers
Platform server
Content Gateways
Elasticsearch cluster
Swarm Storage cluster
Client Network
Private Storage Cluster Network
Internet
Cloud Service



Deployment Planning

- Environment Planning
- Swarm Planning
- VMware Planning
- Registration

Environment Planning

Research and itemize these environment components before deploying any Swarm Storage solution:

Component	Value	Notes
Enclosures (number)	single | dual	Example: Dell PowerEdge M1000e
Storage servers/blades (number)	3 or more	Example: Dell PowerEdge R440

Application servers needing direct cluster access	name, purpose	List all. Example: Enterprise Vault, information governance
Content Gateway (S3) will be used?	yes | no	
DHCP server at deployment site?	yes | no	
Network Time Protocol (NTP) time server(s)	name/IP address	List all. Use of a local NTP server is best practice
Domain Name Service (DNS) server(s)	name/IP address	List all.
LDAP / Active Directory server(s)	name/IP address	List all.
Timezone of deployment	abbrev, offset	Example: CDT, UTC -5
Default gateway for deployment network	name/IP address	
Subnet mask for deployment network		Example: 255.255.255.0

Swarm Planning

Plan for how the Swarm Storage cluster will be configured:

Component	Example	Notes
Name of storage cluster (will be default domain)	defaultdomain.example.com	Must be a DNS fully qualified name format
Default object replicas	reps = 2	
Default erasure-coding scheme	5:2 for objects > 1 MB	
Will cluster be replicated?	yes | no	
Reserved IPs for Caringo management services	IP1, IP2	Each enclosure needs an assigned IP for this use
Will storage be accessed by SSL/TLS?	yes | no	
If yes, do you have an SSL/TLS server certificate?	yes | no	
Is a load balancer or SSL offload system already in place?	yes | no	

Content Gateway will use which identity system?	default (PAM) LDAP Active Directory None (anonymous)	
Operating system to install?	CentOS or RHEL	If using RHEL (Red Hat Enterprise Linux), your site needs a Red Hat license

VMware Planning

If using virtual machines in your deployment, research and itemize the following:

Component	Value	Notes
IP address(es) reserved for the ESXi host machine(s)	IP(s)	Must be reserved from the network where the solution will be deployed.
System name (FQDN) for vCenter Server Appliance management endpoint		Must be reserved and is required for SSL certificate creation by VMware vCenter.
vCenter Server Appliance management endpoint	hostname / DNS	
Will you create a vCenter SSO domain or use an existing?	new | existing	
vCenter SSO domain		
vCenter SSO site name		
iDRAC gateway	IP	For the integrated Dell Remote Access Controller (iDRAC) interface, enter IP addresses to be assigned for iDRAC access
iDRAC subnet	IP	

Registration

Be sure to complete these registrations before proceeding:

1. Register with Caringo Connect: connect.caringo.com
2. Register with Caringo Support: support.caringo.com
3. Register with VMware, if needed: vmware.com

- Capacity Planning
- Hardware Selection
- Cluster Planning

- Deployment Best Practices

Capacity Planning

Following is a high-level view of factors to consider when researching what hardware capacity you will need for your Swarm implementation.

Storage Capacity Factors

Expected Object Count and Average Object Size

- Object count and object size are the primary drivers for capacity planning
- Object count drives storage cluster memory requirements: more objects requires more memory for the cluster's overlay index
- Average object size multiplied by object count provides the logical storage footprint (the amount of content that has been uploaded to the cluster), but it does not account for the space taken by replicas/segments from your protection scheme
- Average object size is the key factor (along with cluster size) for which protection scheme to use (replication vs. erasure coding)

> See Elastic Content Protection.

Choice of Protection Scheme

- Which protection scheme you choose drives the memory requirements for your storage cluster
- Erasure coding (EC) requires more memory than Replication (which uses more space)
- Erasure coding impacts CPU performance requirements (because of calculating parity for erasure coding)
- Required volume footprint is derived from combination of (object count) x (average object size) x (protection scheme overhead)
  - Replication example: (1 million objects) x (1 megabyte/object) x (2 replicas) = 2 TB
  - EC example: (1 million objects) x (1 megabyte/object) x (5:2 EC scheme or 7/5) = 1.4 TB

RAM per Node	16 GB	32 GB	64 GB	128 GB
Storage Node RAM index slots	268M	536M	1073M	2146M
Immutable Objects	268M	536M	1073M	2146M

Mutable Objects	134M	268M	536M	1073M
5:2 Erasure Coded Objects	26M	53M	107M	214M

> See Configuring Content Policies.

Need for High Availability

Knowing what failure scenarios you can and cannot tolerate will help with design optimization:

- A requirement for high availability (HA) drives extra capacity needed to cover more catastrophic disk and server failures
- Designs typically account for either multiple volume or multiple server failure scenarios
- Availability requirements can be simple or complex, and they usually feed back into protection scheme choice

> Best practice
> Whenever cluster used capacity reaches 80%, start expanding the cluster.

Memory for Overlay Index

- Other features may be enabled in a cluster which require more resources in order to properly support them
  - Example: Overlay Index for large clusters (32+ nodes)
- Always consider and account for the resource impact of a given feature/setting before enabling it in your cluster!

> Best practice
> Allow an additional 25% of cluster memory to support its Overlay Index.

Elasticsearch (Search and List)

- Provides ability to search for and list objects based on their metadata
- Always assume full index of object metadata (custom metadata)
- Memory — 64 GB RAM per 1 billion distinct objects
- Disk — 1.5 TB required for 1 billion distinct objects
- Networking — 1 Gb Ethernet minimum
- Server Count: minimum of 3 to 4, for redundancy and performance
- Scale out as needed by adding more Elasticsearch servers

Gateway (including S3)

- Provides reverse proxy into storage with added protocol conversion support (S3) and authentication & authorization policy enforcement
- Best treated with a “scale out” approach (think “web farm” behind a load balancer)
- Underlying engine is Java (Jetty)

- Tuned out of the box to account for large session counts based on field feedback
- Memory/CPU/Disk requirements are light for single Gateway server (4 GB RAM/multi-core x86-64/4 GB Disk)
- Networking should align with choice used for Storage Cluster (for example, if Storage Cluster is using 10 Gb interfaces, use the same for the Gateway servers)

SwarmNFS

- Provides a protocol gateway for NFS clients (NFS v4.1 to SCSP+)
- Resource requirements are primarily driven by level of concurrent write requests
- Best practice: split up differing NFS client workloads across multiple SwarmNFS servers ("scale out")
- Memory/CPU/Disk requirements are somewhat higher than Gateway (recommended baseline of 16 GB RAM/multi-core x86-64/40 GB Disk)
- As with Gateway, networking choice should align with Storage Cluster choice to ensure throughput

FileFly

- Provides a transparent tiering mechanism to move data from Windows or NetApp file servers into a Swarm storage cluster
- Deployments can range from simple "single server" configurations to multi-server/high-availability architectures
- Agent software has a small footprint (minimal servers require 4 GB RAM, x86-64 CPU, 2 GB Disk for logs, etc.)
- Treat as a "scale out" solution to support multiple Windows/NetApp file servers (multiple migration agents, multiple fpolicy servers)
- Make sure the servers under FileFly source management are "close" to Swarm on the network (avoid routing)
- Align network interface choice for FileFly components with those used in Storage Cluster for best throughput/latency characteristics
- Note that sources under FileFly management tend to become "oversubscribed" (i.e., more data associated with the source server exists in Swarm than can be held locally by the source server)
- As a result, capacity planning for the FileFly source servers becomes important when you want to perform a large de-migration from Swarm
- Make sure that you plan for this scenario when assigning storage shares from the source servers to clients

Hardware Selection

- Storage CPU
- Storage Memory
- Storage Drives
- Storage Networking
- Minimum Hardware for Storage
- Production Hardware for Storage
- Hardware for Other Components

Storage CPU

- Swarm Storage supports standard x86-64 CPUs (Intel, AMD)
- Single or multiple sockets supported (and multi-core)

- Recommend use of above CPUs that include AES New Instructions (AES-NI) support
  - used by Swarm for improved performance of Encryption at Rest (EAR)
  - most modern server processors include this as of 2010

Storage Memory

RAM per storage node for the following object capacities:

RAM per Node	16 GB	32 GB	64 GB	128 GB
Storage Node RAM index slots	268M	536M	1073M	2146M
Immutable Objects	268M	536M	1073M	2146M
Mutable Objects	134M	268M	536M	1073M
5:2 Erasure Coded Objects	26M	53M	107M	214M

Notes:

- Memory required is a function of object count, object type and data protection scheme chosen
- Larger clusters need additional memory for the Overlay Index or other features which may require additional resources

Storage Drives

- Direct Attached
- Controllers: SAS or SATA JBOD HBAs (SAS preferred)
- “Hot plug” connector / backplane support
- Disks: “Enterprise Grade”
  - designed for 24x7 continuous duty cycles
  - typically 5 year warranty
  - Examples: Seagate “Exos”, Western Digital “Gold”

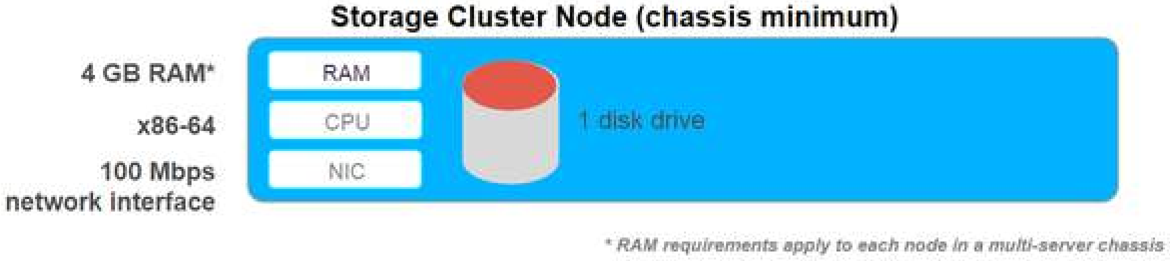
Storage Networking

- Ethernet (with appropriate connector type)
- 1 Gb to 10 Gb (or higher if needed)
- Bonding of multiple ports supported for throughput & redundancy
- including 802.3ad (LAG/LACP) if switch redundancy is required
- Jumbo Frame support
- Typical vendor choices are Intel, Broadcom etc.

Minimum Hardware for Storage

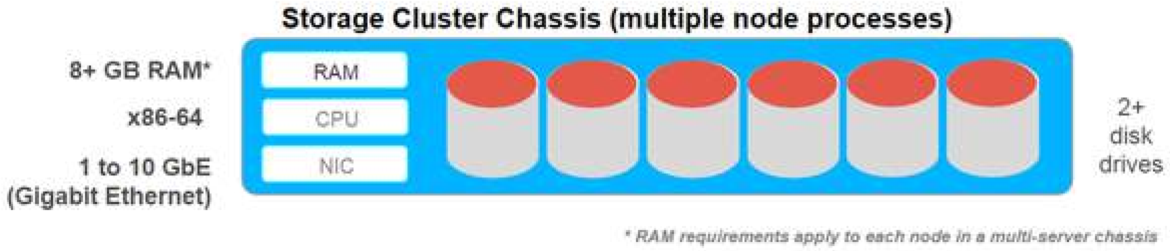
- Appropriate for functional design & testing

- 3 or more nodes (chassis) in a cluster
- Can be deployed as virtual machines (VMware guests)


Storage Cluster Node (chassis minimum)
4 GB RAM*
x86-64
100 Mbps network interface
RAM
CPU
NIC
1 disk drive
* RAM requirements apply to each node in a multi-server chassis

Production Hardware for Storage

- Multi-socket / Multi-core x86-64 CPUs
- “Enterprise Grade” SAS drives
- RAM depends on object counts and other factors
- Minimum of 4 nodes (chassis) in the cluster (scale up / scale out)
- Typically physical servers, but can be virtual machines (VMware)


Storage Cluster Chassis (multiple node processes)
8+ GB RAM*
x86-64
1 to 10 GbE (Gigabit Ethernet)
RAM
CPU
NIC
2+ disk drives
* RAM requirements apply to each node in a multi-server chassis

Hardware for Other Components

Component	Platform Server	Elasticsearch	Content Gateway	SwarmNFS
Purpose	Boot, monitor, manage Storage cluster	Query and list objects in Storage	Protocol and auth/auth gateway to Storage	NFS protocol gateway to Storage
CPU	x86-64 (multi-socket/core, 2 cores)	x86-64 (multi-socket/core)	x86-64 (multi-socket/core)	x86-64 (multi-socket/core, 4+ cores)
Memory	8 GB RAM	64 GB RAM per 1 billion distinct objects	4+ GB RAM	4+ GB RAM (16 GB recommended)
Drive	80+ GB (large clusters: more for logs)	1.5 TB per 1 billion distinct objects	4+ GB plus OS install footprint	40+ GB plus OS install footprint

Network	1 Gb Ethernet	1 Gb Ethernet	1 Gb Ethernet	1 Gb Ethernet (10 Gb heavy traffic)
Servers	1	3 to 4 (for redundancy and performance)	Scale to support client sessions	Scale to support client sessions
Virtualize	Yes (VMware - OVA available)	Yes (VMware)	Yes (VMware)	Yes (VMware)
Notes		Assume full index of object metadata (custom metadata)		Scale RAM and CPU with concurrent writes

Cluster Planning

With Swarm 10's density-friendly architecture, every physical or virtual machine only requires one IP address to run, which simplifies administration. Each Swarm "node" simply refers to the physical or virtual machine that hosts it, because only a single instance of Swarm software runs on it.

When designing your Swarm cluster, observe the following data protection requirements and guidelines:

- Subclusters — All nodes remain in the single, default subcluster unless you manually group them into named subclusters by setting `node.subcluster` across your nodes. Do this if you want Swarm to distribute content according to groupings of machines with a shared failure mode, such as being in the same building in a widely distributed cluster. (Setting `ec.protectionLevel=subcluster` without creating subclusters will cause a critical error and lower the protection level to 'node'.)
- Replication — For data protection reasons, a single Swarm node does not store multiple replicas of an object. If you are using fewer physical machines than are required for your replication scheme, be sure to use a virtualization/containerization technology in order to run multiple Swarm nodes on the same piece of hardware.
- Erasure-coding — Best practice is to use `ec.protectionLevel=node`, which distributes segments across the cluster's physical/virtual machines. Do not use `ec.protectionLevel=subcluster` unless you already have subclusters defined and are sure that you still have enough nodes (machines) to support your specified EC encoding. The lowest level, `ec.protectionLevel=volume`, allows EC writes to succeed if you have a small cluster with fewer than (k+p)/p nodes. See the table below for details.

  > Note
  > Swarm always seeks the highest protection possible for EC segments, regardless of the level you set.

- Small clusters — If you have 10 or fewer Swarm nodes (never use fewer than 3 in production), verify the following settings.
  Important: If you need to change any, do so before upgrading to Swarm 10.
  - policy.replicas — The `min` and `default` values for numbers of replicas to keep in your cluster must not exceed your number of nodes. For example, a 3-node cluster may have only `min=2` or `min=3`.
  - EC encoding — For EC encoding, verify that you have enough nodes to support your cluster's encoding (`policy.ecEncoding`). For EC writes to succeed with fewer than (k+p)/p nodes, use the lower level, `ec.protectionLevel=volume`.
  - Best practice: Keep at least one physical machine in your cluster beyond the minimum number needed.

This allows for one machine to be down for maintenance without compromising the constraint.

- "Cluster in a box" — Swarm supports a "cluster in a box" configuration as long as that box is running a virtual machine host and Swarm instances are running in 3 or more VMs. Each VM boots separately and has its own IP address. Follow the recommendations for small clusters, substituting VMs for nodes. If you have two physical machines, use the "cluster in a box" configuration, but with 3 or more, move to direct booting of Swarm.

ec.protectionLevel	Cluster requirements	Effect
subcluster	>= (k+p)/p subclusters	Requires a subcluster for every p segments.
node (default)	>= (k+p)/p nodes	Requires a node for every p segments. Important When working with limited nodes, adjust your EC encoding to support what you do have. • Given 3 nodes, you can use `3:2` encoding ((3 + 2) ÷ 2 = 3 nodes required), but not `3:1` encoding ((3 + 1) ÷ 1 = 4 nodes required). • Given 4 nodes, you can use `4:2` encoding ((4 + 2) ÷ 2 = 3 nodes required), but not `4:1` encoding ((4 + 1) ÷ 1 = 5 nodes required).
volume	>= k+p volumes	Least protection. Requires k+p volumes.
	< k+p volumes	Unsupportable. EC writes will fail.

Deprecated

The setting `ec.subclusterLossTolerance` has been deprecated and needs to be removed from configurations when upgrading to Swarm 10.

Deployment Best Practices

- Top-Level Planning
- Storage Cluster Best Practices
- Elasticsearch Best Practices
- Gateway / S3 Best Practices
- SwarmNFS Best Practices
- FileFly Best Practices

Following are a collection of best practices and reminders for various stages and areas of your Swarm 9 implementation.

Top-Level Planning

- Ensure you meet all network requirements
- Configure the Switch/VLAN, such as IGMP Snooping and Spanning Tree
- Decide IP assignments, both cluster and client-facing
- Assign Multicast Group
- Create a detailed diagram of your intended implementation (see Use Cases and Architectures)
- For cluster naming and domains, use IANA FQDN format (`cluster.example.com`), and align them with your DNS
- Follow conventions for SSL/TLS certificates
- Plan Authentication/Authorization and your user store (LDAP, AD, PAM, Tokens) (see Content Gateway Authentication)
- Decide how data will be segmented across tenants, domains, and buckets (see Migrating from Traditional Storage)
- Define your policy for client access and data flow; which clients will be used to create/access data?
- Choose your approach to integrate your client s and applications; will there be multiple protocol access to the same data (namespace)?

Storage Cluster Best Practices

- Itemize and account for performance requirements, if any
- Plan for both maintenance (drive replacement, live upgrades) and disruption (server failure, drive failure) scenarios
  - Make sure that protection scheme choices align with available resources
- Select both monitoring and notification approaches
- Capture utilization trends so that you can stay ahead of capacity planning (increasing hardware and licensing level)
- Create a default domain in the cluster that has the same name as the cluster name (this is the “catch all” for enforcing tenancy of objects in the cluster)
- Ensure that all domains in the cluster use IANA FQDN format, as this has ramifications for DNS, Gateway, S3, and SSL+TLS

> See Storage Implementation.

Elasticsearch Best Practices

- Plan to “scale out” similarly to Swarm Storage
- For best performance and redundancy in production, start out with four ES servers
- Allow no Elasticsearch server to go beyond 64 GB of physical memory (this affects Java max heap and performance)
- To optimize listing and query performance, use SSD drives
- Locate Elasticsearch servers on the same subnet as Storage Cluster (avoid routing to Swarm nodes)
- When performing upgrades, be sure to read the release notes for Storage Cluster re: associated Elasticsearch changes which may be necessary

- Always use the Elasticsearch packages that are bundled with the Swarm version you are deploying

> See Elasticsearch Implementation.

Gateway / S3 Best Practices

- Gateway serves as a “scale out, lightweight reverse proxy” to your object storage
- Place multiple Gateway servers behind the load balancer
- Perform SSL/TLS off-load at the load balancer layer
- Make sure that Gateway servers have unfettered access to Storage Cluster and Elasticsearch nodes
- For best performance, place Gateway servers on the Storage Cluster network
- Make sure that Gateway is provided access to LDAP/AD targets that are “network close” (as few hops as possible) and in good working order
- Monitor concurrent session count for Capacity Planning; heavy S3 request activity may mean that you need additional Elasticsearch resources

> See Content Gateway Implementation.

SwarmNFS Best Practices

- Stateless Protocol Translator from NFSv4 to Swarm (SCSP)
- Be sure to run the latest Swarm version for best performance, Swarm 9.5 or greater
- Scale out vs. export count (memory) and exports that exhibit large concurrent access activity
- Make sure that SwarmNFS deployment planning aligns with authentication and authorization approach for Swarm Storage (Anonymous / Single User / Session Token)
- Make sure that NFS clients can use NFSv4 (other NFS versions not supported)
- For best behavior, clients should mount SwarmNFS exports using the “noatime” flag and with a “timeo” setting of 9000

> See SwarmNFS Deployment.

FileFly Best Practices

- When configuring FileFly, use named servers (DNS, FQDN) rather than IP addresses, so that future server migrations will be easier
- When installing the FileFly plugins, enable both header options, Include metadata HTTP headers and Include Content-Disposition, which allows full metadata capture (such as for creating Collections from FileFly data)
- If possible, deploy FileFly using Gateway (aka CloudScaler) rather than Direct to Swarm
  - Gateway allows for authenticated access and data segmentation / policy protection of FileFly data
  - Gateway also supports SSL/TLS encapsulation of data in transit
- With Scrub tasks (which cleans Swarm of data no longer associated with a FileFly source), ensure that the grace period aligns with your overall backup policy
- After performing any data migration tasks, always be sure to run a “DrTool from Source” task
  - Running the tool is necessary to ensure up-to-date recovery of stubs, which may have been accidentally deleted
- FileFly can be sensitive to network throughput, so keep the associated source and target systems as “close” on

the network as possible, and use the highest bandwidth available

- Make note of the location of the FileFly logs, for troubleshooting

Deployment Process

Following is a high-level view of the nature and order of tasks you will need to perform for a full-stack Caringo Swarm implementation.

> Important
> Before starting these tasks, be sure to complete Deployment Planning in consultation with Caringo Sales and Support.

- Phase 1: Prepare Environment
- Phase 2: Platform Server and Storage Cluster
- Phase 3: Elasticsearch
- Phase 4: Content Gateway
- Phase 5: Caringo Clients
- Phase 6: Post-installation

Phase 1: Prepare Environment

The work to prepare the environment must be completed before adding any Caringo components:

- Rack and stack hardware designated for Swarm, replacing and upgrading as needed. (See Hardware Setup.)
- Upgrade firmware to latest versions:
  - All servers
  - All disk controllers
  - All disk drives
- Configure networking and switches (see Network Infrastructure), including the following:
  - VLAN configuration
  - IGMP snooping disabled (or IGMP querier implemented)
- Configure IPMI management.
- Provide access for the storage cluster to phone home. (See Cluster Health Report.)
- Ensure the servers and base operating systems meet the Swarm system requirements. (See Hardware Requirements for Storage and Hardware Requirements for Elasticsearch Cluster.)
- Configure IPMI (remote server management)
- Complete licenses and agreements
  - Obtain any needed storage capacity and capability licenses from Caringo. (See Licensing Swarm.)
  - If installing or updating RHEL, a user in your organization must register the Red Hat license and accept the EULA.
  - Accept the Caringo EULA (via connect.Caringo.com).

Phase 2: Platform Server and Storage Cluster

Swarm Platform Server is installed and configured first, so that it can install Storage nodes on the designated hardware.

- Install Platform Server. (See Platform Implementation.)
- Configure Platform Server to integrate with your environment.
- Configure Platform Server to boot the current version of Swarm Storage.
- Boot the Storage nodes and configure the cluster-wide settings. (See Configuring the Nodes.)
- Install the Swarm Storage UI. (See Swarm Storage UI Installation.)
- Verify that the storage cluster is operational: read, write, and delete test objects

Phase 3: Elasticsearch

Install and configure an Elasticsearch cluster on designated hardware, providing the Storage cluster its search and metrics capabilities.

- Base install your chosen operating system (RHEL/Centos 7.4).
- Install Elasticsearch nodes on designated hardware. (See Elasticsearch Implementation.)
- Configure Elasticsearch based on Caringo recommendations. (See Configuring Elasticsearch.)
- Create a Search feed to populate the Elasticsearch metadata index. (See Viewing and Editing Feeds.)
- Configure Elasticsearch Curator and Swarm Metrics. (See Installing Swarm Metrics.)

Phase 4: Content Gateway

Install and configure Content Gateways, which provide the primary access to the Storage cluster.

- Base install your chosen operating system (RHEL/Centos 7.4).
- Install the Content Gateway. (See Content Gateway Implementation.)
- Install the Content UI. (See Content UI Installation.)
- Configure basic gateway setup for verification and initialization of primary domain. (See Gateway Configuration and Configuring Swarm Storage for Gateway.)
- Verify that the Gateway is operational: read, write, and delete test objects via the Content UI, S3, and SCSP.
- Create the initial domains, with policy definitions. (See Gateway Access Control Policies.)

Phase 5: Caringo Clients

As fits with your implementation plan, extend access to Swarm storage by installing one or more Swarm client applications, such as the following:

- SwarmNFS
- FileFly
- Caringo Drive

> Optional Swarm Components
> These are optional Swarm components, each with separate distribution packaging and licensing.

Phase 6: Post-installation

- Conduct performance measurement and tuning.
- Test and debug your third-party/custom applications and integration.
- Train your administrators and staff.

Network Infrastructure

This section describes how to set up your Swarm network infrastructure in your corporate enterprise.

- Understanding Swarm in the Network
- Setting up the Swarm Network
- Setting up the Network Services
- Setting up PXE Booting
- Network Devices and Priority
- Proxying the Swarm Admin Console
- IGMP Snooping
- Tuning Network Performance

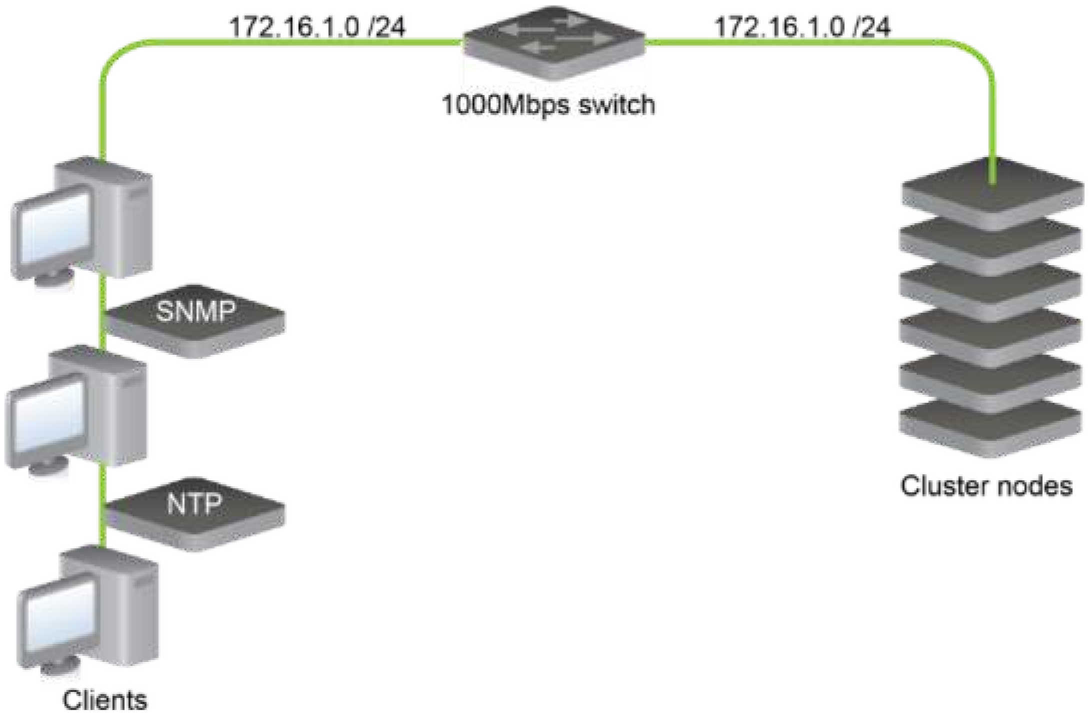
Understanding Swarm in the Network

This section provides a high-level overview of setting up a storage cluster in your network.

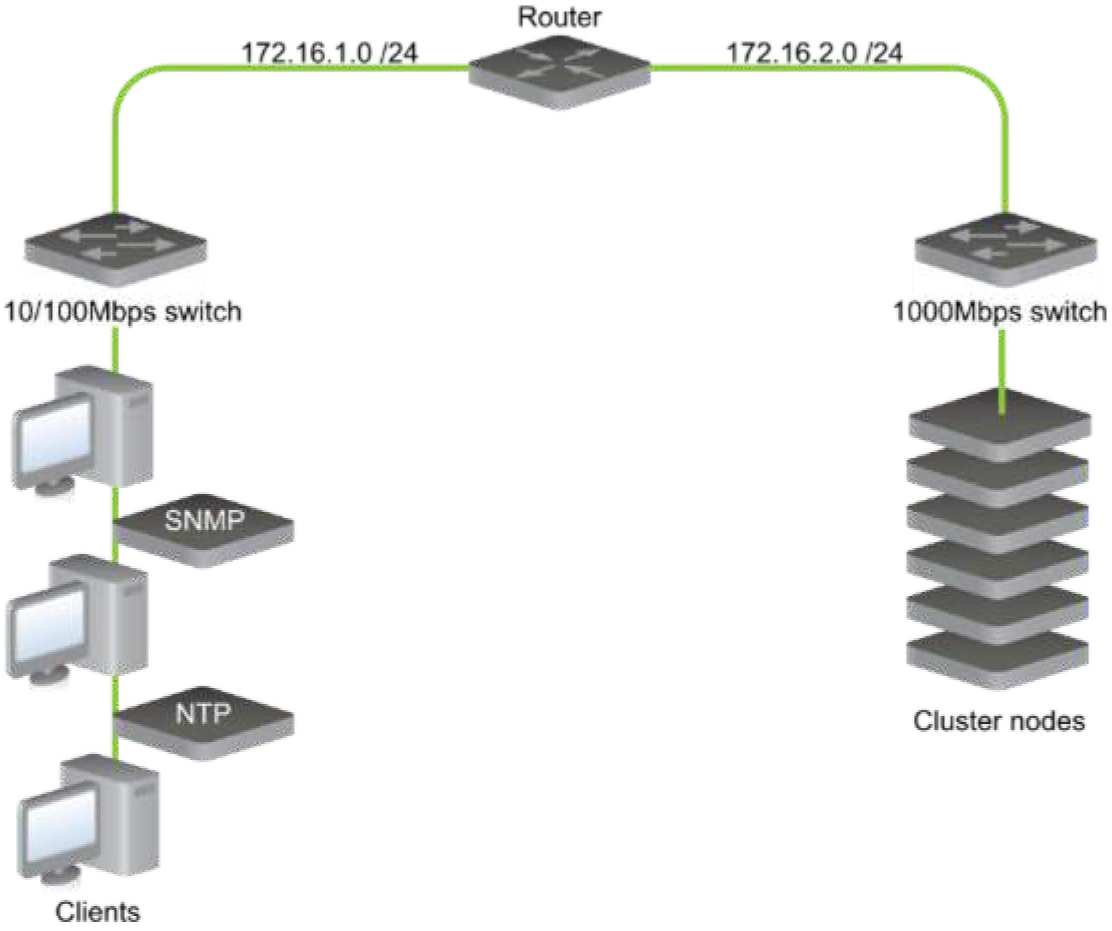
- Sample Networks
- Layer 3 Switching and Routing
- Switching Hardware
- Internet Deployments

Sample Networks

The following illustration shows a network where the storage cluster nodes and clients are located in the same subnet using a 1000 Mbps switch. This network is easy to set up and requires basic hardware components, but does not offer any traffic separation between the Swarm nodes and the remaining network.



The next illustration shows a network where the storage cluster nodes and clients are located on separate subnets using a router.



Layer 3 Switching and Routing

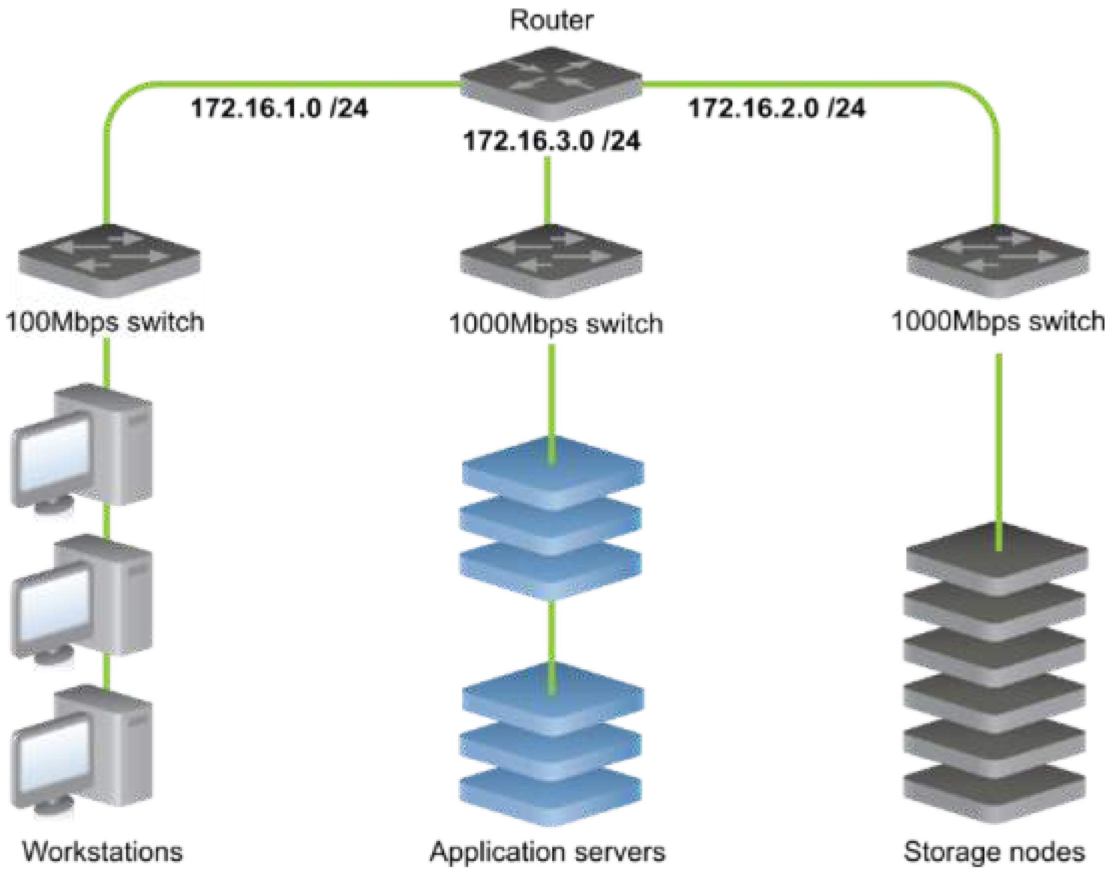
A router or an Open Systems Interconnection (OSI) layer 3 switch routes network packets between subnets. A router segregates network traffic by filtering packets based on the targeted subnets. Separating the subnets provides the Swarm nodes with a stable network bandwidth so the multicast and unicast traffic between each node in your storage network does not interfere with the systems and devices in your corporate network.

Switching Hardware

If your client workstations are configured with 100 Mbps network interface controllers (NICs) or cannot effectively use more than 100 Mbps of bandwidth, connecting these systems to 1000 Mbps Ethernet switches may not be cost-effective. In this case, consider connecting these workstations to a separate Ethernet switch that supports the slower bandwidth speed.

When selecting Ethernet switching hardware, remember that many client workstations are configured with 100 Mbps NICs, and it may not be cost-effective to connect these workstations to 1000 Mbps ports. Additionally, the operating systems and applications running on these workstations might be unable to use more than 100 Mbps of bandwidth effectively.

The following network architecture has the client workstations, application servers, and Swarm storage nodes isolated on switches that support their maximum bandwidth speeds.



Using advanced switches that support multiple routing capabilities, you can isolate your network segments as Virtual LANs (or VLANs) on the same device.

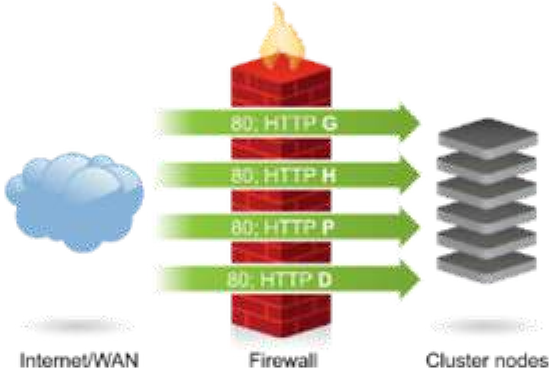
To provide high availability when a switch fails, design your Swarm storage network subnet to incorporate redundant switches. When Swarm nodes are connected to multiple network switches, a redundant path provides uninterrupted data communications between the nodes if a switch fails for any reason. Deploying Swarm in a multiple switch environment (or switched fabric) requires planning and an understanding of your corporate IT structure.

To provide effective data communications between each switch port, make sure that the bandwidth in your switched fabric exceeds the port speed on each switch. For information about proprietary software or implementing link aggregation in your Swarm network, contact your switch provider.

Internet Deployments

When deploying any service on the Internet or within an extensive enterprise wide area network (WAN), network security is a top priority. In these situations, install a firewall or filtering router in front of the storage cluster nodes to control the types of traffic and requests that access your cluster nodes.

The following illustration shows a firewall that allows requests on TCP port 80, the default Simple Content Storage Protocol (SCSP) port. If the SCSP port value set in the storage cluster node or cluster configuration file is not port 80, reset the firewall TCP port to match the value in the configuration file.



If the firewall can examine HTTP request content or traffic on OSI layer 7 (the Application layer), additional configuration is required to only allow your supported SCSP methods.

- To present a cluster as a read-only device to external clients, block the POST and DELETE requests to prevent updates to the cluster.
- To prevent client access to the Node Status window in the Swarm Admin Console, configure the firewall to deny "GET /" requests to the cluster nodes.
- To prevent unauthorized access to the Swarm Admin Console, block Internet access to the Swarm Admin Console port (default TCP port 90) and the SNMP port (UDP port 161). Wide area networks (WANs) may require additional restrictions to prevent access to specific administrative networks or workstations.
- To minimize the client impact of hardware failures, deploy devices in redundant pairs when adding security devices such as firewalls into the network architecture.

Setting up the Swarm Network

Client applications must be able to initiate TCP connections with all nodes in a storage cluster using the designated access port, which is typically port 80. In a storage cluster, the nodes must be able to communicate with each other using UDP, TCP, and multicast communication protocols.

This section describes how to set up a storage cluster in a standard TCP/IP networking environment.

Network Communications

These are the required and optional network communications used in a storage cluster:

Communication	Type	Port	Requirement
Legacy Console	TCP	90	Pulls information from a Swarm node into the legacy Admin Console.
Swarm UI	TCP	91	Pulls information from a Swarm node into the Swarm Management UI.
SCSP	TCP	80	Required Important Although unnamed objects are nearly impossible to guess by UUID, the metadata on those objects in Elasticsearch may reveal information about them. Secure port 80 from calls that might be probing for such metadata.

Health report	HTTPS	443	Required for proactive support from Caringo. See Cluster Health Report.
DHCP	UDP	67	Recommended
DNS	UDP	53	Optional
mDNS	UDP	5353	Optional
Multicast	UDP	7000	Required
NTP	UDP	123	Required
Logging	UDP	514	Required for support
SNMP	UDP	161	Required for support
TFTP	UDP	69	Optional

Setting up the Network Services

This section describes how to set up the network services for your storage cluster.

> Platform Server
> If you use Platform Server, skip this section: your network services are set up.

- Setting up NTP for time synchronization
- Setting up DHCP for IP address administration
- Setting up DNS for name resolution
- Preparing for domains
- Setting up a Syslog Server for Critical Alerts
- Setting up SNMP for monitoring
- Setting up network load balancing
- Setting up the network interfaces

Setting up NTP for time synchronization

The Network Time Protocol (NTP) server provides time synchronization between the cluster nodes, which is critical for many Swarm components. For best results, configure multiple NTP servers in close proximity to your cluster. For example, you can use the NTP Pool Project's continental zones, which are pools of NTP servers.

One or more trusted NTP servers, such as dedicated hardware solutions on your internal network or publicly available NTP servers, are required in your storage cluster. This configuration is required, ensuring that the internal clocks in all nodes are synchronized with each other.

If trusted NTP servers are available, you can add these servers to your cluster by adding their IP addresses or host names in the `network.timeSource` parameter located in the node configuration files. The parameter value is a list of one or more NTP servers (either host names or IP addresses) separated by spaces. For example, to add a second NTP server IP address, use the following syntax:

```
network.timeSource = 10.20.40.21 10.20.50.31
```

To add an NTP server host name, the node must be able to resolve host names using a DNS server. Use this syntax:

```
network.timeSource = ntp1.example.com ntp2.example.com
```

> See Configuring an External Time Server.

NTP 3.0

NTP 3.0 included a design limitation that causes the time value to wrap in the year 2036. If the BIOS clock in a cluster node is set beyond this wrap point, NTP cannot correct the time. Before you boot Swarm in your cluster, ensure that the BIOS clocks in all nodes are set to a year prior to 2036. This issue was resolved in NTP 4.0.

If the cluster nodes cannot access an external or internal NTP server, see Configuring a Node without NTP.

Setting up DHCP for IP address administration

The Dynamic Host Configuration Protocol (DHCP) server provides IP addresses to the cluster nodes and other devices that are enabled as DHCP clients. While Swarm nodes are not required to have static IP addresses to discover and communicate with each other, administrators might find it easier to manage and monitor a cluster where each node receives a predetermined IP address.

To configure this option using DHCP:

1. Map the Ethernet media access control (MAC) address of each node to a static IP address.
2. Configure your DHCP server to provide each node with an IP address for each of these:
   - network mask
   - default gateway
   - DNS server

Setting up DNS for name resolution

The Domain Name Service (DNS) is used to resolve host names into IP addresses. While DNS is not required for Swarm nodes to communicate with each other, DNS can be very useful for client applications to reach the cluster. If you use named objects, DNS is one method you can use to enable access to objects over the Internet.

Best practice

Although client applications can initiate first contact with any node in the storage cluster – even choosing to access the same node every time – best practice is for the node of first contact to be distributed evenly around the cluster.

For example, you can:

- Define multiple DNS entries ("A" or "CNAME" records) that specify the IP address for the same Swarm first contact node.
- Use multiple IP addresses for a DNS entry to create a DNS round-robin that provides client request balancing.

> See your DNS software documentation for how to use "A" records and "CNAME" (alias) records.

Swarm requires a DNS server to resolve host names in the configuration file. For example, you can add a host name to the NTP list or the log host (such as ntp.pool.org) for name resolution. The DNS server needs to be set in the Swarm configuration file. In contrast, applications must resolve Swarm domain names to find the storage cluster. These unique requirements will most likely be addressed using different DNS servers.

The following example shows the entries in the Internet Systems Consortium (ISC) BIND DNS software configuration file for three node IP addresses tied to one name.

```
Swarm 0 IN A 192.168.1.101
      0 IN A 192.168.1.102
      0 IN A 192.168.1.103
```

In this example, the Time To Live (TTL) value for each of the records in the round-robin group is very small (0-2 seconds). This configuration is necessary so that clients who cache the resolution results will quickly flush them. This process allows the first contact node to be distributed and allows a client to move quickly to another node if the first contact node is unavailable.

> Best practice
> Applications should implement robust mechanisms such as Zero Configuration Networking for distributing the node of first contact and skipping failed nodes, but an administrator can use DNS to assist with simpler applications.

Preparing for domains

To allow clients to access named objects over the Internet, enable incoming HTTP requests to resolve to the correct domain. (A cluster can contain many domains, each of which can contain many buckets, each of which can contain many named objects.) Cluster and domain names should both be Internet Assigned Numbers Authority (IANA) compatible host names, such as cluster.example.com.

For example, a client application can create an object with a name such as:

```
cluster.example.com/marketing/photos/ads/object-naming.3gp
```

In this example, cluster.example.com is the domain name, marketing is the name of a bucket, and photos/ads/object-naming.3gp is the name of an object. Set up your network so the host name in the HTTP request maps correctly to the object's domain name. The cluster name is not required.

To enable clients to access a named object:

1. Set up your hosts file to map domain names to IP address(es) of the first contact node.

    - For a Linux system, configure the /etc/hosts file.
    - For a Windows system, configure the %SystemRoot%\system32\drivers\etc\hosts file.
      Example of a configured hosts file:

```
192.168.1.111 cluster.example.com
192.168.1.112 vault.example.com
```

2. Define multiple DNS entries ("A" or "CNAME" records) that identify the IP address(es) of the first contact node in the storage cluster. This process creates a DNS round-robin that provides client request load balancing.
    - For help setting up DNS for Swarm, see Setting up DNS for Name Resolution.
    - For information about setting up your DNS server, see your DNS software documentation.

Setting up a Syslog Server for Critical Alerts

You must set up a syslog server to capture critical operational alerts from the nodes in a storage cluster. The server captures messages sent by the Swarm nodes on UDP port 514.

> See Configuring External Logging on configuring an rsyslog server and the log.host and log.level parameters used to send Swarm messages to a syslog server.

Setting up SNMP for monitoring

Swarm provides monitoring information and administrative controls using the Simple Network Management Protocol (SNMP). Using an SNMP console, an administrator can monitor a storage cluster from a central location.

Swarm uses an SNMP management information base (MIB) definition file to map SNMP object identifiers (OIDs) to logical names. The MIB can be located in one of two locations, depending on your configuration:

- If your cluster nodes boot from a Platform Server, the aggregate MIB for the entire cluster is located at /usr/share/snmp/mibs.
- If your cluster nodes do not boot from a Platform Server, the MIB is located in the root directory of your Swarm software distribution.

> See Using SNMP with Swarm.

Setting up network load balancing

Although the Swarm Storage Cluster nodes interact with client applications using the HTTP communication protocol, the nodes operate differently from traditional web servers. As a result, placing storage nodes behind an HTTP load balancer is usually an unnecessary configuration. A properly configured load balancer can add value-added services like SSL off-load and centralized certificate management.

During normal operations, a storage node routinely redirects a client to another node within the cluster. When this process occurs, the client must initiate another HTTP request to the redirected node. Any process that virtualizes the storage node IP addresses or attempts to control the nodes connected to the client will generate communication errors.

Setting up the network interfaces

Gigabit Ethernet or faster NICs provide the recommended 1000 Mbps data communications speed between your storage cluster nodes. Swarm automatically uses multiple NICs to provide a redundant network connection.
To implement this feature, connect the NICs to one or more interconnected switches in the same subnet.

> See Switching Hardware.

Setting up PXE Booting

- Setting up the DHCP server for PXE booting
- Configuring PortFast on your switch ports
- Configuring the TFTP server
- Setting up a configuration file server
- Disabling monitor power-saving activation

This section describes how to boot a cluster from the network using the Intel Preboot Execution Environment (PXE) specification. This booting process (commonly referred to as network booting) is supported by most NICs. PXE is one way to boot your storage cluster nodes.

> Platform Server
> If you use Platform Server, skip this section: your network booting is set up.

- To enable nodes to boot from a USB flash drive, see Initializing a Storage Cluster.
- To enable nodes to boot using a configuration file server, see the section below.
- To enable nodes to PXE boot, perform these steps:
  a. Configure your DHCP server with next-server and filename parameters.
  b. Configure PortFast on the switch ports leading to the storage cluster nodes.
  c. Configure the TFTP server with PXE bootstrap, configuration, and Swarm files.
  d. Set up the nodes' BIOS configurations for network booting.

> Requirement
> To prevent PXE boot failures, be sure to increase the size of the initrd RAM disk to 160MB on your PXE boot server. This does not apply if you use Platform Server.

Setting up the DHCP server for PXE booting

> Warning
> Swarm can erase all non-Swarm data on hosts that boot accidentally from the network. When you set up your DHCP server, verify that it provides network booting information to the correct network hosts only.

The following example shows the configuration lines from the Internet Systems Consortium (ISC) DHCP server that is commonly available on UNIX systems. As shown below, the next-server parameter defines the IP address of the Trivial File Transfer Protocol (TFTP) server and the filename parameter to define the bootstrap loader program to download.

```
group {
   next-server 172.16.1.10;
   filename "/pxelinux.0";
   # Hosts allowed to network boot into Swarm
   host clusternode1 { hardware ethernet 00:90:cb:bf:45:26; }
   host clusternode2 { hardware ethernet 00:90:b2:92:09:e4; }
   host clusternode3 { hardware ethernet 00:90:0d:46:7a:b4; }
   }
```

In this example, the Swarm nodes are explicitly defined by MAC address to prevent Swarm from initiating an unattended boot by other servers or workstations.

Configuring PortFast on your switch ports

PortFast is a switch port configuration parameter that enables a port to bypass the listening and learning Spanning Tree states so the port will immediately forward traffic.

If you connect a storage cluster node to a network switch, ensure that PortFast is configured on the switch ports leading to each node. Otherwise, the extended time delay can prevent netboot from delivering the Swarm image to a PXE-enabled node in a timely manner.

Configuring the TFTP server

The TFTP server transfers configuration or boot files between systems in a local environment. After you configure your DHCP server, configure your TFTP server to load the Swarm software onto the cluster nodes.

To set up your TFTP server:

1. Install and configure TFTP server software on the boot server.
2. Create the /tftpboot directory hierarchy.
3. Copy the kernel and fsimage files to the /tftpboot/profiles/castor directory.

> See DHCP and Boot Server Redundancy, below.

Installing and configuring TFTP

TFTP server software is available in both free and commercial packages. UNIX distributions commonly include TFTP server software with their standard setup. For example, the tftp-hpa package for UNIX can integrate with Swarm. You can also obtain source code from the Linux Kernel Archives website located at kernel.org/pub/software/network/tftp.

TFTP server software is also available as a binary package in many Linux distributions.

Creating the tftpboot directory hierarchy

After you install the TFTP server, configure the server to access the network boot file directory. This directory is typically labeled /tftpboot because TFTP is almost exclusively used for booting network devices.

A sample template is included in the samples/NetworkBoot directory of your Swarm software distribution.

Copying kernel and fsimage

The Swarm software distribution media includes the kernel and fsimage files, which contain the Swarm embedded operating system. Copy these files to the tftpboot/profiles/castor directory on the TFTP server so they will load onto each Swarm node during bootup.

After you copy the directory template and the Swarm software files, the tftpboot directory on the TFTP server should contain these files:

File name	Description
`tftpboot/pxelinux.0`	Boot loader program
`tftpboot/profiles/castor/fsimage`	Swarm software
`tftpboot/profiles/castor/kernel`	Swarm operating system kernel
`tftpboot/pxelinux.cfg/default`	PXELINUX configuration file

> See the documentation and ZIP file in the samples/Network-Boot directory on the Swarm distribution media for help with using the PXELINUX boot loader.

DHCP and boot server redundancy

When you set up your DHCP server, configure both a primary and secondary DHCP server. This configuration eliminates a single point of failure if one of the servers goes offline for any reason.

- To set up the ISC DHCP daemon for redundancy, see "Failover with ISC DHCP" at madboa.com/geek/dhcp-failover.
- To provide redundancy at the network booting layer, you can use your primary and secondary DHCP servers as TFTP servers.
  When you set up your DHCP servers, set the next-server parameter in each server to specify their own IP address. When the primary or secondary DHCP server answers a DHCP query, it will also handle the PXE boot.
- To prevent any network interruptions, ensure that the TFTP boot servers are located in the same broadcast domain (or VLAN) as the Swarm nodes or enable a DHCP relay server on the VLAN.

Setting up a configuration file server

Platform Server

If you use Platform Server, skip this section: your centralized configuration is set up.

Swarm supports centralized node configuration files on an HTTP or FTP server. This method allows you to boot from a network or a standard USB flash drive. A centralized configuration file server simplifies storage cluster administration by supporting configuration file updates and providing a method to group similar node configurations together.

To implement a configuration file server, set the value of the castor_cfg kernel configuration parameter to a URL that targets the configuration list file, as described below.

PXE boot example

This is an example PXELINUX configuration file located in the tftpboot/pxelinux.cfg directory on the TFTP boot server.

```
default profiles/castor/kernel
append initrd=profiles/castor/fsimage
ramdisk_size=160000 root=/dev/ram0
 castor_cfg=http://172.16.1.200/castor/cfg-list.txt
```

USB boot loader example

This is an example section of the syslinux.cfg located in the root directory on the USB flash drive.

```
label normal
 kernel kernel
 append initrd=fsimage ramdisk_size=160000 root=/dev/ram0
  castor_cfg=http://172.16.1.200/castor/cfg-list.txt
```

Configuration list file example

The castor_cfg kernel configuration parameter specifies a file that contains a list of URLs for all the configuration files that will be loaded by a Swarm node. Swarm configuration files are evaluated in the order in which they are listed in the configuration list file.

Although Swarm configuration settings can be defined multiple times, only the last definition is used. By redefining the settings, you can layer configuration files so that they contain generally applicable values for a cluster, a group of similar nodes, and values specific to one node.

Example of URLs in a configuration list file:

```
http://172.16.1.200/castor/cluster.cfg
http://172.16.1.200/castor/subcluster.cfg
http://172.16.1.200/castor/testnode.cfg
```

Each of the configuration files in the list file uses the same format as the Swarm node.cfg file.

> See the /caringo/node.cfg.sample in the Swarm software distribution.
> See Managing Configuration Settings.

Disabling monitor power-saving activation

To disable the monitor power-saving feature from activating while connected to a Swarm storage node, add the following kernel option to the APPEND line in either the syslinux.cfg file on the Swarm USB key or in the PXE boot configuration file.

When enabled, this parameter tells the kernel to stop blanking the console:

```
consoleblank=0
```

This feature defaults to 10 minutes. A value of 0 disables the blank timer. Listed below are examples.

PXE boot example

This is a PXELINUX configuration file from the tftpboot/pxelinux.cfg directory on the TFTP boot server with the console power saver disabled.

```
default profiles/castor/kernel
append initrd=profiles/castor/fsimage consoleblank=0 ramdisk_size=160000
root=/dev/ram0
 castor_cfg=http://172.16.1.200/castor/cfg-list.txt
```

USB boot loader example

This is a section of the syslinux.cfg contained in the root directory on the USB flash drive with the console power savings disabled.

```
label normal
 kernel kernel
 append initrd=fsimage consoleblank=0 ramdisk_size=160000
root=/dev/ram0
  castor_cfg=http://172.16.1.200/castor/cfg-list.txt
```

Network Devices and Priority

By default, all Swarm Ethernet network adapters are encapsulated into a redundant bond interface where one device is active and the remaining devices are backups. In this default mode, the first network device is the preferred device. You can override this behavior by:

- Excluding a network adapter from use, such as an intelligent platform management interface (IPMI) card.
- Changing the preferred network adapter for network load management.
- Bonding multiple adapters together for increased throughput (such as NIC teaming).

To override the network adapter, configure the switch ports to the appropriate mode, such as link aggregation or 802.3 ad. Swarm supports all Linux bonding driver supported modes.

To override the Swarm default network device settings, edit one of the following boot configuration files:

- syslinux.cfg if the node is booting from the USB flash drive
- pxelinux.cfg if the node is booting from the network

In the configuration file, a kernel parameter named castor_net is included in the append clause. castor_net lets you specify both the bonding mode for the adapters as well as a comma-separated ordered list of the network devices that Swarm can use. In this scenario, the first device in the list is the preferred interface that will be used whenever it is online.

> Important
> The list of network devices must use the adapter names assigned by Swarm. To locate the current list of adapter names and MAC addresses, boot the node from your Swarm USB drive to access the System Menu.

Swarm assigns device names to adapters based on a sorted list of MAC addresses. Device names begin with eth followed by an integer starting at 0. Adding network hardware can change the assignment order.

Below are some examples of assigning device names in the configuration file. The other portions of the append clause are abbreviated for clarity. Note the trailing colon after the bonding mode:

```
append initrd=... castor_net=active-backup:eth1,eth0
append initrd=... castor_net=balance-rr:eth0,eth1
append initrd=... castor_net=802.3ad:eth1,eth0
append initrd=... castor_net=eth1,eth2
append initrd=... castor_net=802.3ad:
```

Proxying the Swarm Admin Console

Administrators running Swarm on a private, protected network may choose to allow clients on the external network to view the Swarm console without providing them access to the nodes themselves on the private network. You can do this by proxying the console from a privileged server that straddles the internal and external networks.

For example, from a server running Apache HTTP Server , the following rewrite rule for URLs could be applied using the mod_rewrite module:

```
<Location /storage/>
 RewriteEngine On RewriteRule ./storage/([ /]+)/(.)$ http://$1:90/$2
[P,L]
</Location>
```

IGMP Snooping

- Disabling IGMP snooping
- Enabling an IGMP querier
- Node logging of IGMP snooping

Disabling IGMP snooping

Managed switches may implement IGMP snooping to direct multicast traffic to their ports. The purpose of IGMP snooping is to block unnecessary multicast traffic from hosts who are not interested in the traffic. The switch will only forward multicast traffic out to ports where it heard an IGMP join message within a configurable time (typically around 5 minutes).

However, Swarm nodes should exist in their own private VLAN so that there are no other hosts in the broadcast domain. You should disable IGMP snooping from the Swarm nodes' VLAN because there is no benefit to having IGMP snooping configured in a VLAN that only includes Caringo products.

If you cannot disable IGMP snooping from the VLAN, you need to configure an IGMP querier for your cluster's multicast group(s).

> Warning
>
> Enabling IGMP snooping without an IGMP querier results in nodes that cannot communicate with each other.

> See your router documentation for details on enabling an IGMP querier.
>
> For more information on IGMP Snooping, see RFC 1112, RFC 2236, and RFC 3376.

Enabling an IGMP querier

When a Swarm node joins a cluster, it sends an initial set of unsolicited join requests for its configured multicast group. At that point, all Swarm nodes are visible from the Swarm Admin Console of all other nodes. IGMP queriers periodically send another query to see if there are any hosts still interested in the multicast group. As required by the IGMP RFCs, Swarm nodes will not send addition unsolicited join requests. If there is no querier for that multicast group in the network, the switch will stop forwarding multicast traffic for that particular group out of that particular switch port when the switch timer for that multicast group runs out. After the timeout, all Swarm nodes appear to be unable to contact each other because the router did not send a query to prompt a subsequent join by the Swarm node.

Some switches are configured to act as IGMP queriers in an IPv4 network for multicast group memberships, but other switches are not unless configured appropriately. Since multicast routing is not configured by default on all switches, an IGMP querier may not exist on your network unless you have specifically configured one.

If you do not have a querier available in your network, the Swarm can be configured to perform this function. When configured to do so, Swarm will elect a node in the cluster to act as an IGMP querier, ensuring that multicast membership queries are sent so cluster operation can continue. The Swarm IGMP querier will have no impact on networks that do not have IGMP Snooping enabled.

To enable an IGMP querier on the Swarm cluster itself, set it via the SNMP MIB entry networkIGMPTimeout and enable the network.igmpTimeout configuration parameter in Swarm.

Node logging of IGMP snooping

To help identify cluster networks where IGMP snooping is enabled without an IGMP querier, a node will log a critical error on the Swarm Admin Console and in the syslog, recommending administrators check for the presence of IGMP snooping. The node does this when these conditions occur:

- It could previously multicast to a node.
- It can no longer multicast to that node.
- It can still unicast to that node.

By default, Swarm uses IGMPv2 responses to host membership queries. The igmpVersion parameter can be used to force the use of version 1, 2, or 3.

Tuning Network Performance

- Running Benchmarks

Network tuning is beyond the scope of Swarm Storage, but it is an important part of your implementation. You can contact Caringo for assistance with your network tuning.

By default, Linux networking is configured to optimize reliability, not performance, which becomes apparent with GbE adapters: the kernel’s send/receive buffers, TCP memory allocations, and packet backlog are generally too small. With gigabit Ethernet, tuning can significantly improve performance.

> Important
>
> For best results, run benchmarks on non-Swarm nodes in your network, using tools such as FTP to do data transfers to measure performance. After you optimize settings for non-Swarm systems in your network environment, you can then apply those settings to the Swarm nodes in your cluster.

For each GbE controller, Intel includes a README for Linux that outlines recommendations. Be sure to refer to the documentation supplied by the manufacturer of your controller.

These are the most important performance-tuning changes to make (per Intel's ixgb driver documentation), in order of greatest impact:

1. Enable jumbo frames on your local hosts and switches.
2. Use sysctl to tune the Linux kernel settings.

> See Performance tuning: Intel 10-gigabit NIC, which incorporates the tuning recommendations cited in the Intel documentation.

Running Benchmarks

> Best practice
>
> Change one setting at a time, running benchmarking tools (such as `iperf` and `netperf`) to determine the impact of that change.

Before starting any benchmarks, temporarily disable `irqbalance` and `cpuspeed` on your Linux-based test client(s) to maximize network throughput and allow the best results:

Benchmark prep

```
service irqbalance stop
service cpuspeed stop
chkconfig irqbalance off
chkconfig cpuspeed off
```

Enable Jumbo Frames

Before proceeding, run a benchmark so that you can confirm the performance impact.

To enable jumbo frames, you increase the value of the maximum transmission unit (MTU). TCP uses the MTU to determine the maximum size of each packet in any transmission.

> Important
>
> Update this setting across all components of your Swarm implementation, including all switches and nodes.

<table>
<tr><td>Network switches</td><td>To update the value to your interface config, replace “eth2” with your interface name. See your switch manufacturer's instructions on how to change MTU size.

Caution
Before you change the default value, verify that the node's network interfaces and all other network hardware support the selected MTU; otherwise, the nodes might not be able to replicate objects or communicate.</td></tr>
<tr><td>Swarm nodes</td><td>To update the value for your Swarm storage nodes, update the network.mtu setting in each node.cfg file and reboot. (This value is not stored in the persisted Settings object because it is specific to a given node.)
<table>
<tr><th>Setting</th><th>Default</th><th>Value</th><th>Type</th><th>Description</th></tr>
<tr><td>network.mtu</td><td>1500</td><td>9000</td><td>int</td><td>(Node-specific) In bytes. Sets the maximum transmission unit (MTU) that Swarm accepts. Set to a higher value to use jumbo frames.</td></tr>
</table>
</td></tr>
<tr><td>Other components</td><td>These include such components as test clients and Content Gateway. See the instructions for the component's operating system on how to change MTU size.</td></tr>
<tr><td>VMs</td><td>If the other components are VMs, follow VMware instructions on how to change MTU for ESXi hosts and guests.</td></tr>
</table>

Change sysctl Settings

Before proceeding, run a benchmark so that you can confirm the performance impact.

Next, optimize the core memory settings in the Linux kernel.

> RHEL 7
>
> In RHEL 7, system tunables are set in the /etc/sysctl.d/ directory, and they might be specified in more than one configuration file in this directory. The ordering logic determines which is used, so ensure that your changes are not being overridden. See https://access.redhat.com/solutions/800023.

Gateway components and clients

Here is a modified /etc/sysctl.conf, which can be applied as sysctl changes. Swarm-specific recommendations are grouped at the end. Follow your operating system's recommendations and instructions for modifying sysctl settings.

Modified /etc/sysctrl.conf

```
# -- tuning -- #
# Increase system file descriptor limit
fs.file-max = 65535

# Increase system IP port range to allow more concurrent connections
net.ipv4.ip_local_port_range = 1024 65000

# -- 10gbe tuning from Intel ixgb driver README -- #

# turn off selective ACK and timestamps
net.ipv4.tcp_sack = 0
net.ipv4.tcp_timestamps = 0

# memory allocation min/pressure/max.
# read buffer, write buffer, and buffer space
net.ipv4.tcp_rmem = 10000000 10000000 10000000
net.ipv4.tcp_wmem = 10000000 10000000 10000000
net.core.rmem_max = 524287
net.core.wmem_max = 524287
net.core.optmem_max = 524287

# Caringo-specific recommended values:
net.ipv4.tcp_mem = 134217728 134217728 134217728
net.core.rmem_default = 134217728
net.core.wmem_default = 134217728
net.core.netdev_max_backlog = 300000
```

These kernel updates are hardware/machine specific, so they are not saved to your cluster persistent settings.

Important
Swarm will fail to start if an I/O error occurs while reading a sysctl setting.

Swarm storage nodes

There are per-chassis settings for network tuning that are available for you to set in the node.cfg file and can apply dynamically via SNMP. For sysctl-like and other buffer settings, it is possible to change them via SNMP, but these values are not stored in the persisted Settings object because they are specific to a given chassis.

Storage Setting	SNMP Name	Recommended	Type	Description
sysctl.deviceWeight	deviceWeight	256	int	Value of /proc/sys/net/core/dev_weight.
sysctl.tcpMem	tcpMem	134217728 134217728 134217728	str	Value of /proc/sys/net/ipv4/tcp_mem, in form 'min default max'.
sysctl.coreRMemDefault	rMemDefault	134217728	int	Value of /proc/sys/net/core/rmem_default.
sysctl.coreWMemDefault	wMemDefault	134217728	int	Value of /proc/sys/net/core/wmem_default.
sysctl.netdevMaxBacklog	netdevMaxBacklog	300000	int	Value of /proc/sys/net/core/netdev_max_backlog.
cip.readBufferSize		33554432	int	(Node-specific) In bytes. The size of the multicast UDP socket read buffer.

Set Buffer Size

Before proceeding, run a benchmark so that you can confirm the performance impact.
Swarm supports a limited number of settings that can be tuned, such as buffer size.

Setting	Value	Type	Description
network.wmemMax	262144	int	Maximum value of wmem, ≥ 16384
network.rmemMax	262144	int	Maximum value of rmem, ≥ 87380
network.rxQueueLength	0	int	Value of ethtool -G ethX rx. 0 is unset, leaving kernel default

Hardware Setup

This section addresses how best to configure the hardware devices in your storage cluster.

- Hardware Requirements for Storage
- Hardware Requirements for Elasticsearch Cluster
- Hardware Booting
- Hot Swapping and Plugging Drives
- Local Area Replication with Subclusters
- Multipath Support

Hardware Requirements for Storage

This section describes the hardware requirements for implementing a storage cluster in your corporate enterprise and the best practices for maintaining it.

Note
Swarm installs and runs on enterprise-class (not consumer-grade) x86 commodity hardware.

Caution
To ensure high availability and fail over in the event of a node failure, configure your cluster with a minimum of three nodes.

- Virtualization
- Minimum Requirements
- Recommended Requirements
- Memory Sizing Requirements
- Supporting Erasure Coding
- Supporting High-Performance Clusters
- Balancing Resources
- Selecting Hard Drives
- Mixing Hardware

Virtualization

Swarm storage nodes can run in a VM environment. Swarm currently supports VMware/ESXi. Contact sales for more information and guidance.

Best practice
Enable volume serial numbers on any virtual machines that house Swarm storage nodes (set disk.EnableUUID=TRUE).

Minimum Requirements

The following table lists the minimum hardware requirements for a storage cluster. Because Swarm nodes are designed to run using lights-out management (or out-of-band management), they do not require a keyboard, monitor, and mouse (KVM) to operate.

Component	Requirement
Node	x86 with Pentium-class CPUs
Number of nodes	Three (to ensure adequate recovery space in the event of node failure)
Switch	Nodes must connect to switches configured for multicasting
Node boot capability	USB flash drive or network drive
Network interfaces	One 100 Mbps Ethernet NIC with one RJ-45 port *

Hard drives	One hard drive
RAM	2 GB
NTP server	To synchronize clocks across nodes

* To be compatible, all nodes in a cluster must use the same speed network interface.

Recommended Requirements

The following table lists the recommended hardware requirements for a storage cluster.

Component	Requirement
Node	x86 with Intel Xeon or AMD Athlon64 (or equivalent) CPUs
Number of nodes	Three or more
Switch	Nodes must connect to switches configured for multicasting
Log server	To accept incoming SysLog messages
Node boot capability	USB flash drive or network drive
Network interfaces	Two Dual Gigabit Ethernet NICs with two RJ-45 ports for link aggregation (NIC teaming) Important Variable network speeds for different nodes are not a supported configuration. For example, a node with a 100 Mbps network interface card must not be in the same cluster with a node that has a 1000 Mbps card.
Hard drives	One to four standard non-RAID SATA hard drives
RAM	4 GB
NTP server	To synchronize clocks across nodes

Although Swarm runs on a variety of x86 hardware, this table lists the recommended base characteristics for a storage cluster. Cluster performance improves exponentially by adding additional systems with more intensive CPU hardware and additional memory.

What hardware is best for you depends on your storage and performance requirements, so ask your sales representative for hardware recommendations specific to your needs.

Memory Sizing Requirements

Review the following sections for factors that will influence how you size memory and erasure coding, as well as how you configure Swarm.

How RAM affects storage

The storage cluster is capable of holding the sum of the maximum object counts from all of the nodes in the cluster. How many objects can be stored on a node depends both on the node's drive capacity and the amount of its system RAM.

The following table shows estimates of the maximum possible number of replicated objects (regardless of size) that you can store on a node, based on the amount of RAM in the node, with the default 2 replicas being stored. Each replica takes one slot in the in-memory index maintained on the node.

Amount of RAM	Max. immutable objects	Max. alias or named objects
4 GB	33 million	16 million
8 GB	66 million	33 million
12 GB	132 million	66 million

How the Overlay Index affects RAM

Larger clusters (those above 32 nodes by default) need additional RAM resources to take advantage of the Overlay Index.

To store the same number of reps=2 object counts above and utilize the Overlay Index, increase RAM as follows:

- Immutable unnamed objects: 50% additional RAM
- Alias or named objects: 25% additional RAM

Smaller clusters and larger clusters where the Overlay Index is disabled do not need this additional RAM.

> See Configuring the Overlay Index.

How to configure for small objects

Swarm allows you to store objects up to a maximum of 4 TB. However, if you store mostly small files, configure your storage cluster accordingly.

By default, Swarm allocates a small amount of disk space to store, write, and delete the disk's file change logs (journals). In typical deployments, this default amount is plenty because the remainder of the disk will be filled by objects before the log space is consumed.

However, for installations writing mostly small objects (1 MB and under), the file log space can fill up before the disk space. If your cluster usage focuses on small objects, be sure to increase the configurable amount of log space allocated on the disk before you boot Swarm on the node for the first time.

The parameters used to change this allocation differ depending on the software version in use.

> Tip
> Contact Support for guidance on setting the optimal parameters for your configuration.

Supporting Erasure Coding

Erasure coding conserves disk space but involves additional calculations and communications, so you should anticipate its impact on memory and CPU utilization.

> Takeaway
> Erasure coding uses about half the space of replication but it requires more RAM and other resources.

CPU	In general, plan to support EC with more and faster CPU cores.
Memory	Scale up for larger objects: In general, larger objects require more memory to manage the erasure sets. How many EC objects can be stored on a node per GB of RAM depends on the size of the object and the encoding you specify in the configuration. The erasure-coding manifest takes two index slots per object, regardless of the type of object (named, immutable, or alias). Each erasure-coded segment in an erasure set takes one index slot. Larger objects can have multiple erasure sets, so you would have multiple sets of segments. In k:p encoding (integers for the data (k) and parity (p) segment counts), there are p+1 manifests (up to the ec.maxManifests maximum). For 5:2 encoding, that would mean 3 manifests. For example, with the default segment size of 200 MB and a configured encoding of 5:2, • 1-GB object: (5+2) slots for segments and (2+1) for manifests = 10 index slots • 3-GB object: 3 sets of segments @ 10 slots each = 30 index slots Increase for Overlay Index: Larger clusters (above 32 nodes by default) need additional RAM resources to take advantage of the Overlay Index. For erasure-coded objects, allocate 10% additional RAM to enable the Overlay Index.
Network	Network use by the requesting SAN is an important factor in EC performance. The requesting SAN must orchestrate k+p segment writes (for an EC write) or k segment reads (for an EC read). Because these segment reads and writes must occur at the same rate, the slowest one will slow the overall request. Consequently, when a cluster experiences a lot of EC requests, nodes can play different roles and slower nodes can affect multiple requests.

> See Erasure Coding EC

Supporting High-Performance Clusters

For the demands of high-performance clusters, Swarm benefits from faster CPUs and processor technologies, such as large caches, 64-bit computing, and fast Front Side Bus (FSB) architectures.
To design a storage cluster for peak performance, maximize these variables:

- Add nodes to increase cluster throughput – like adding lanes to a highway
- Fast or 64-bit CPU with large L1 and L2 caches

> Important
> If the cluster node CPU supports hyper-threading, be sure to disable this feature within the BIOS setup to prevent single-CPU degradation in Swarm.

- Fast RAM BUS (front-side BUS) configuration
- Multiple, independent, fast disk channels
- Hard disks with large, on-board buffer caches and Native Command Queuing (NCQ) capability
- Gigabit (or faster) network topology between all storage cluster nodes
- Gigabit (or faster) network topology between the client nodes and the storage cluster

Balancing Resources

For best performance, try to balance resources across your nodes as evenly as possible. For example, in a cluster of nodes with 7 GB of RAM, adding several new nodes with 70 GB of RAM could overwhelm those nodes and have a negative impact on the cluster.

Because Swarm is highly scalable, creating a large cluster and spreading the user request load across multiple storage nodes significantly improves data throughput, and this improvement increases as you add nodes to the cluster.

> Tip
>
> Using multiple replicas when storing objects in the cluster is an excellent way to get the most out of Swarm, because each copy provides redundancy and improves performance.

Selecting Hard Drives

Selecting the right hard drives for your storage nodes improves both performance and recovery, in the event of a node or disk failure. When selecting drives, follow the guidelines below. For an in-depth overview of drive characteristics, download the Intel white paper Enterprise class versus Desktop class Hard Drives .

> Best practice
>
> In order to ensure maximum compatibility, consult with Caringo support prior to purchasing additional hardware.

Drive type	Enterprise-level	The critical factor is whether the hard drive is designed for the demands of a cluster. Enterprise-level hard drives are rated for 24x7 continuous-duty cycles and have time-constrained error recovery logic that is suitable for server deployments where error recovery is handled at a higher level than its on-board controller. In contrast, consumer-level hard drives are rated for desktop use only; they have limited-duty cycles and incorporate error recovery logic that can pause all I/O operations for minutes at a time. These extended error recovery periods and non-continuous duty cycles are not suitable or supported for Swarm deployments.

Reliability	Rated for continuous use	The reliability of hard drives from the same manufacturer will vary, because the drive models target different intended use and duty cycles: • Consumer models targeted for the home user typically assume that the drive will not be used continuously. As a result, these drives do not include the more advanced vibration and misalignment detection and handling features. • Enterprise models targeted for server applications tend to be rated for continuous use (24x7) and include predictable error recovery times, as well as more sophisticated vibration compensation and misalignment detection.
Performance	Large on-board cache Independent channels Fast bus	You can optimize the performance and data throughput of the storage sub-system in a node by selecting drives with these characteristics: • Large buffer cache — Larger, on-board caches improve disk performance. • Independent disk controller channels — Reduces storage bus contention. • High disk RPM — Faster-spinning disks improve performance. • Fast storage bus speed — Faster data transfer rates between storage components, a feature incorporated in these types: • SATA-300 • Serial Attached SCSI (SAS) • Fibre Channel hard drives Use of independent disk controllers is often driven by the storage bus type in the computer system and hard drives. • PATA — Older ATA-100 and ATA-133 (or Parallel Advanced Technology Attachment [PATA]) storage buses allow two devices on the same controller/cable. As a result, bus contention occurs when both devices are in active use. Motherboards with PATA buses typically only have two controllers. If more than two drives are used, some bus sharing must occur. • SATA — Unlike PATA controllers, Serial ATA (SATA) controllers and disks include only one device on each bus to overcome the previous bus contention problems. Motherboards with SATA controllers typically have four or more controllers. Recent improvements in Serial ATA controllers and hard drives (commonly called SATA-300) have doubled the bus speed of the original SATA devices.

Recovery	Avoid highest capacity	You can improve the failure and recovery characteristics of a node when a drive fails by selecting drives with server-class features yet that are not the highest capacity. • Higher capacity means slower replication. When choosing the drive capacity in a node, consider the trade-off between the benefits of high-capacity drives versus the time required to replicate the contents of a failed drive. Larger drives take longer to replicate than smaller ones, and that delay increases the business exposure when a drive fails. • Delayed errors mean erroneous recovery. Unlike consumer-oriented devices, for which it is acceptable for a drive to spend several minutes attempting to retry and recover from a read/write failure, redundant storage designs such as Swarm need the device to emit an error quickly so the operating system can initiate recovery. If the drive in a node requires a long delay before returning an error, the entire node may appear to be down, causing the cluster to initiate recovery actions for all drives in the node — not just the failed drive. • Short command timeouts mean less impact. The short command timeout value inherent in most enterprise-class drives allows recovery efforts to occur while other system drives continue to support system drive access requests by Swarm.
Controllers and RAID	JBOD, not RAID Controller-compatible	Best practice: Check with Caringo before investing in new equipment, both for initial deployment and for future expansion of your cluster. Caringo can help you avoid problems not only with drive controller options but also with network card choices. • Evaluate controller compatibility before each purchasing decision. • Buy controller-compatible hardware. As a rule, the more types of controllers in a cluster, the more restrictions you face on how volumes can be moved. Study these restrictions, and keep this information with your hardware inventory. • Avoid RAID controllers. Always choose JBOD over RAID when specifying hardware for use in Swarm. RAID controllers are problematic, for these reasons: • Incompatibilities in RAID volume formatting • Inability of many to hot plug, so you lose the ability to move volumes between machines • Problems with volume identification (drive lights) • HP Proliant systems with P840 RAID controllers - Swarm performance degrades when this card runs in HBA mode. For best performance, use only single raid 0s per drive, but be aware that you cannot hotplug volumes in this system.

Firmware	Track kernel mappings	Be sure to keep track of controller driver to controller firmware mappings in the kernel that ships with Swarm Storage. This is particularly important when working with LSI-based controllers (which are the majority), because a mismatch between driver and firmware in LSI's Fusion MPT architecture can introduce indeterminate volume behavior (such as good drives reporting errors erroneously due to a driver mismatch).

Mixing Hardware

Swarm greatly simplifies hardware maintenance by making drives independent of their chassis and their drive slots. As long as your drive controllers are compatible, you are free to move drives as you need. Swarm supports a variety of hardware, and clusters can blend hardware as older equipment fails or is decommissioned and replaced. The largest issue with mixing hardware is incompatibility among the drive controllers.

For best results with mixing hardware, follow these guidelines:

Track controllers	When you administer the cluster, monitor your hardware inventory with special attention to the drive controllers. Some RAID controllers, for example, reserve part of the drive for controller-specific information (DDF). Once a volume is formatted for use by Swarm, it must be used with a chassis having that specific controller and controller configuration. To save time and data movement, many maintenance tasks involve physically relocating volumes between chassis. Use the inventory of your drive controller types to easily spot when movement of formatted volumes is prohibited due to drive controller incompatibility.
Test compatibility	To determine controller compatibility safely, test outside of your production cluster. 1. Set up two spare chassis, each with the controller being compared. 2. In the first chassis, format a new volume in Swarm. 3. Move the volume to the second chassis and watch the log for error messages during mount or for any attempt to reformat the volume. 4. Retire the volume in the second chassis and move it back to the first. 5. Again, watch for errors or attempts to reformat the volume. 6. If all goes well, erase the drive using dd and repeat the procedure in reverse, where the volume is formatted on the second chassis. If no problems occur during this test, you can confidently swap volumes between these chassis within your cluster. If this test runs into trouble, do not swap volumes between these controllers.

Hardware Requirements for Elasticsearch Cluster

- Scaling Elasticsearch
- Hardware Best Practices
- RAM for Elasticsearch
- Disk Usage for Search
- Optimizing Disk I/O for ES
- Optimizing Disaster Recovery for ES

Your Elasticsearch cluster supports both Swarm searches and Historical Metrics. The Swarm Feeds mechanism populates the metadata search servers that run the Elasticsearch (ES) software.

> See Elasticsearch Implementation and Installing Swarm Metrics.

This software requires one or more servers running RHEL/CentOS 7 Linux. Although Elasticsearch runs on other Linux platforms, we currently provide support and testing for these versions. Only the Elasticsearch version provided with the Swarm distribution is supported.

> See the Elasticsearch project website for more information about Elasticsearch.

Scaling Elasticsearch

The hardware platform for the Elasticsearch servers can scale from one virtual machine to multiple physical machines.

> Best practice
> Elasticsearch recommends not exceeding 200 million documents (unique objects) per ES node. Staying within the limit for number of documents per node maximizes performance.

Scale	ES nodes	Max documents	Swarm nodes	Metrics retention	Notes
Minimum	1	200M	3		ES 5.6: Run script to set replicas to zero (see below). Do not use without a robust ES backup strategy
Minimal	2	400M	≤ 50	≤ 90 days	Not recommended; lower performance than Basic
Basic	3	600M	≤ 100	≤ 120 days	
Large	5	1000M	≤ 200	≤ 240 days	
Very large	8	1600M	> 200	≤ 365 days	

> For details about the Metrics templates, see Installing Swarm Metrics.

Replicas in ES 5.6 — Elasticsearch 5.6 no longer allows you to configure the number of replica shards for new indices in the `elasticsearch.yml` file. Instead, it provides a default replica value of 1, which works for most installations. However, if you are configuring an Elasticsearch cluster with only one Elasticsearch data node, then all your indices will have yellow status, because the primary and replica shards should be hosted on separate nodes. In that case, configure your indices to have zero replicas, using the script provided:

```
/usr/share/caringo-elasticsearch-search/bin/configure_replicas.py -r 0
-e <ES-SERVER>
```

To view the complete options for changing the number of replicas on existing indices, use the help command:

```
/usr/share/caringo-elasticsearch-search/bin/configure_replicas.py --help
```

Replicas in ES 2.3.3 — By default, Elasticsearch allocates 5 primary shards and one set of replicas per index. With three or more ES servers, this provides high availability. With five or more servers, you can better spread the transactions across multiple servers, providing additional processing power.

- Swarm Search indices use the default 5 shard/1 replica scheme. If you need higher availability and query throughput, configure additional replicas by editing the "index.number_of_replicas" property in the elasticsearch.yml file.
- Swarm Metrics indices are configured with 1 shard and one additional replica per index. Additional replicas can be configured by editing the "index.number_of_replicas" property in the elasticsearch.yml file. If more shards are desired, contact Support for help with that configuration.

Hardware Best Practices

Following are overall best practices, with hardware recommendations from Elasticsearch:

- Provision the machines with CPUs that have at least 4 cores and 64 GB memory. Between faster processors or more cores, choose more cores.
- Choose SSD drives, to boost performance.
- If using hard disk drives (which do not handle concurrent I/O as well as SSDs), do the following:
  - Select high-performance server drives.
  - Use RAID 0 with a writeback cache.
  - Set `index.merge.scheduler.max_thread_count` to 1, to prevent too many merges from running at once.

    ```
    curl -X PUT <ES_SERVER>:9200/<SWARM_SEARCH_INDEX>/_settings \
     -d '{ “index”: {“merge.scheduler.max_thread_count”: 1}}'
    ```
- As with your storage cluster, choose similar, moderate configurations, for balanced resource usage.

RAM for Elasticsearch

RAM is key for Elasticsearch performance. Use these guidelines as a basis for capacity planning:

- 64 GB RAM per machine is optimal (recommended by Elasticsearch).
- Dedicate half of total RAM to the Java Virtual Machine (JVM) that runs Elasticsearch, but do not exceed 31 GB, for best performance.
- For ES 5.6.x, disable swapping of the Elasticsearch image. (For ES 2.3.3, allow in-memory caching of all shards on the server.)

You can achieve optimal performance by adding adequate RAM in your ES servers to store all database shards in memory. Take steps to disable or mitigate swapping. If memory page swapping begins to occur on an ES server, it will impact Elasticsearch performance.

Important
When monitoring your ES servers, watch for sustained increases in page swapping and disk I/O, which might mean that you need to add RAM to an ES server or deploy additional servers to offset the load.

Disk Usage for Search

The storage on the Elasticsearch servers is used to persist the shards of the Swarm Search and Swarm Metrics indices. Follow these guidelines for capacity planning for the Swarm Search indices.

- Assumption: 2 KB average per metric sample × 7 metrics indices × 4 samples/hour × 120 days data retention ≈ 161 MB per Swarm node being indexed
- Baseline metadata to support listing only: 150 GB per 200 million objects
- Full metadata to support ad-hoc searching: 300 GB per 200 million objects
- Custom metadata: if you index a large amount of custom metadata, allocate additional storage in proportion

These are objects, not replicas: how many Swarm replicas a Swarm object has is irrelevant to the ES servers. No matter how many replicas of an object exist in the cluster, there will be only one metadata entry for the object.

Tip
Do not confuse this with the RAM-based Overlay Index that each storage node maintains, which depends on the total number of replicas in the cluster.

Optimizing Disk I/O for ES

Elasticsearch makes heavy use of drives, so higher throughput means more stable nodes. Follow these Elasticsearch guidelines for optimizing disk I/O:

- Use SSDs. SSDs will boost performance. With SSDs, ensure that your OS I/O scheduler is configured correctly.
- Use RAID 0. Striped RAID will increase disk I/O, at the expense of potential failure if a drive dies. Do not use mirrored or parity RAIDS, because replicas provide that functionality. Alternatively, use multiple drives and allow Elasticsearch to stripe data across them via multiple path.data directories.
- Do not use remote-mounted storage, such as NFS or SMB/CIFS; the latency will hurt performance.
- Avoid virtualized storage, such as EBS (Amazon Elastic Block Store). Even SSD-backed EBS is often slower than local instance storage.

Optimizing Disaster Recovery for ES

Elasticsearch clustering has been specifically designed to mitigate the impact of hardware and power failures, so that you do not experience long delays from refreshing the search data. How much you should invest to optimize your hardware depends on how important metadata search and querying is to your organization and how long these features can be offline while Elasticsearch rebuilds its data.

These are principles for making your configuration more disaster-proof:

- Do not use and rely on just a single Elasticsearch server. This makes your search and metrics capabilities vulnerable to disruption, and it risks too little capacity to support all your search and metrics needs.
- For power failure protection, deploy enough Elasticsearch servers to survive multiple server failures and distribute them across different power sources.

- If your cluster is divided into subclusters to match your power groups, then set up Elasticsearch with multiple nodes that are spread equally among the subclusters. This strategy improves survivability of a power group failure.

Hardware Booting

The Swarm node hardware needs to be configured to boot either from the USB flash drive or a PXE network (which is the default method for the Platform Server).

> See Setting up PXE Booting.

Platform Server
If you use Platform Server, skip this section: your hardware is set up.

Booting from USB

If your cluster boots from a USB device, your hardware may identify the device as either a hard drive or a removable device. Ensure that the USB device appears at the top of the boot priority order so the system will boot from the USB drive before any other devices on the node.

Because all internal drives are typically used for storage, the BIOS may prevent you from booting to a hard drive. Make sure that the boot priority of your hard drives is lower than your USB flash drive.

Keyboard Errors

A keyboard attached to the cluster node is not required during normal Swarm operations.

Note
You may need to configure the cluster node BIOS to avoid keyboard errors during a boot.

Hot Swapping and Plugging Drives

Administrators can insert a drive into a running node as long as the server hardware supports this function. Replacing failed drives (hot swapping) or adding additional drives (hot plugging) is supported without a server reboot.

When you insert a new unformatted drive, Swarm recognizes, formats, and mounts the drive as a new volume. When you insert a Swarm-formatted drive into the same node or into a different node, the drive continues to function as a volume without data loss. If the formatted drive was previously retired, the volume remains retired.

No manual configuration or intervention is needed. Messages display in logs and in the Swarm Admin Console to indicate that a drive was inserted or removed.

Requirements for Hot Plugging

Note
Not all hardware supports hot plugging in Swarm correctly. To determine if your hardware is supported, contact your account representative.

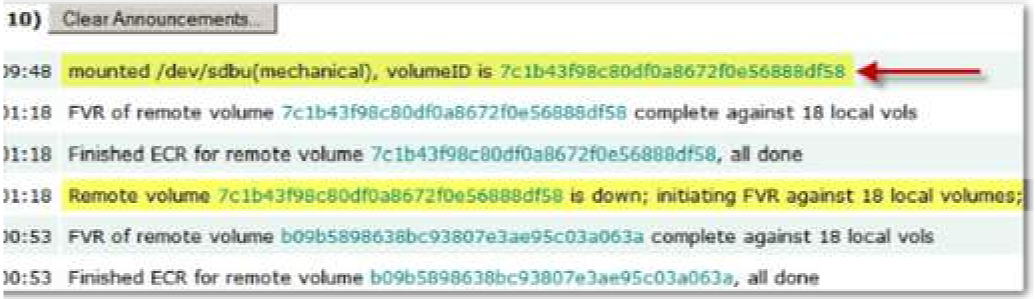
- The configuration option disk.volumes must be set to all.
- To use a drive with a RAID controller, JBOD/pass-through mode must be supported and enabled. Contact

support for details.

- Drives must not be configured in RAID.
- Any virtual machines that house Swarm storage nodes must enable disk UUIDs (set disk.EnableUUID=TRUE).

Guidelines for Hot Swapping

- To determine if hot swapping succeeded, count the total drives (status OK) across all node processes and make sure that equals what you had before you pulled the drives. Be aware that the drive might not be assigned to the same node process that was handling it before you moved it.
- If SNMP is slow in its updates, expect that a drive that is plugged back in should at least show up in the first node process on the machine; that said, the drive add algorithm will attempt to keep the volume assignments balanced across node processes.
- Check the drive identification lights: The drive identification light is automatically enabled if a drive cannot mount when hot-plugged into a system or fails at boot time.
- When pulling a live, good drive (not retired or disabled due to error count), expect to see "noise" in the syslog about failed volume recovery (FVR) starting. Look for these announcements:
  - FVR has completed or been cancelled on the hot-swapped drive.
  - The hot-swapped volumeID is mounted and recognized by the assigned node process.



```
9:48  mounted /dev/sdbu(mechanical), volumeID is 7c1b43f98c80df0a8672f0e56888df58
1:18  FVR of remote volume 7c1b43f98c80df0a8672f0e56888df58 complete against 18 local vols
1:18  Finished ECR for remote volume 7c1b43f98c80df0a8672f0e56888df58, all done
1:18  Remote volume 7c1b43f98c80df0a8672f0e56888df58 is down; initiating FVR against 18 local volumes;
0:53  FVR of remote volume b09b5898638bc93807e3ae95c03a063a complete against 18 local vols
0:53  Finished ECR for remote volume b09b5898638bc93807e3ae95c03a063a, all done
```

Local Area Replication with Subclusters

Local area replication (LAR) allows you to create logical separations in a storage cluster to define storage distribution strategies. These logical separations cause Swarm to attempt to create the greatest logical spread between an object's replicas by moving them into separate subclusters. Examples where LAR subclusters are useful include:

- Splitting a cluster based on location (data cabinet, building wing)
- Grouping nodes based on common infrastructure (network, power)

For example, if you have data centers located in separate wings of your building and you want to have copies of stored content exist in both locations in case of a partial building loss, consider splitting your cluster based on location. However, if you want to group your cluster nodes by shared network switches, a common power distribution unit (PDU), or a common electrical circuit within a rack, grouping the cluster nodes based on a common infrastructure can be another option.

> Note
> Multi-server subclusters are a special case. Even though each of its nodes is assigned a chassis-level default subcluster, you can override it to configure a different subcluster.

Network requirements. The network connections between LAR subclusters must have the same speed and latency characteristics as the connections between the nodes. Additionally, all nodes must be in the same broadcast domain such that they are able to send data directly to all other nodes in the cluster and receive the multicast traffic sent from anywhere in the cluster.

> Warning
>
> Minimize moving nodes to new subcluster assignments, which can cause high cluster activity while content is redistributed to conform to the new subcluster assignments.

Space requirements. When you retire a volume, make sure that sufficient space exists in the LAR subcluster that contains the retiring volumes if you want the separation to persist. Because Swarm must maintain the correct number of replicas in the subcluster, retiring a volume without sufficient space can be problematic. For example, Swarm might create all replicas on the other side of the subcluster, filling up that side of the subcluster.

> See node.subcluster in the Settings Reference.

Multipath Support

Understanding Multipath

Within Linux, Device Mapper Multipathing (DM-Multipath) provides I/O failover and load balancing for storage devices. Host servers use multiple paths of a redundant network to provide continuous availability and higher bandwidth connectivity with their devices. DM-Multipathing routes around path failures and balances the load across the paths.

- The problem – Higher-end hardware is offering DM-Multipath support, which provides failover protection through data path redundancy (via multiple cables). Without multipath support, Swarm counts the drives in such a chassis twice because it sees them over both cables (that is, if a chassis has 60 drives, Swarm sees 120 volumes to manage).
- The solution – With multipath support, Swarm can handle redundant cables as one path. This handling prevents Swarm from adding the same physical drive through different volume names. Swarm monitors and updates its multipath mappings for all hot-plug events, such as adding a physical drive, a cable on the multipath device, or a new multipath chassis. With multipath, Swarm makes dynamic accommodation whenever redundant cables fail or are hot plug added or removed:
  - When a second cable is gained or lost on a multipath chassis, no Swarm volumes are affected.
  - When the only cable is gained or lost, all Swarm volumes on that chassis are affected.

> Caution

Changing the multipath configuration setting requires an immediate cluster reboot and renames all of the drives.

How multipath assigns names

The most visible effect of multipath support is how it maps volumes to new names:

- Enabled: With multipath enabled, Swarm assigns every drive in your cluster a multipath name in a pattern like /dev/dm-x, even if none of the existing drives support multipath. Swarm manages the mapping between the multipath name and the underlying drive name.

  Tip
  If Swarm shows names without the prefix dm- (which refers to DM-Multipath), then multipath is not enabled in your cluster.

- Disabled: With multipath disabled, every drive in your cluster keeps its volume name in a pattern like /dev/sdx, even if it is a multipath drive.

  Caution
  Without multipath, Swarm cannot manage redundant connections.

How multipath validates drives

Swarm has validations to prevent physical drives from being mounted more than once and to keep disk controller errors from inconsistently mapping multipaths:

- During startup and hot-plug events, Swarm checks disk volume headers to ensure that a physical drive is not mounted through multiple paths and that it is only referred to through a multipath volume name.
- Swarm compares multipath mappings through different command sequences to ensure that they each provide the same data.
- Swarm partitions drives into sets with the same volume header and compares them to multipath mappings to verify that mappings and partitions are mutually consistent.

Configuring Multipath

You control multipath by configuring a pair of options in the settings file:

- disk.enableMultipath
- disk.volumes

The disk.enableMultipath setting accepts integers, with 0 meaning false and all other values meaning true.

Recommended settings

By default, multipath is disabled, so that you can carefully plan when you will enable it and reboot your cluster:

```
disk.enableMultipath = 1  # Enable multipath; disabled by default
disk.volumes = all        # Multipath vols assigned to nodes by Swarm
```

If you need to specify named volumes (with or without except), the best practice is to disable multipath:

```
disk.enableMultipath = 0  # Disable multipath
disk.volumes = /dev/sda /dev/sdb
```

> Important
> Hot plugging is only supported for disk.volumes = all.

When to disable multipath

Do not disable multipath on the assumption that this would boost performance. Any performance differences would be negligible.

For best results, review your situation with Support and enable multipath unless Support advises against it. Having multipath enabled allows Swarm to handle the drives in a multipath, multi-channel chassis correctly and seamlessly, should any be added later.

Support might advise you to disable multipath (disk.enableMultipath=0) in a few cases:

- You are not using multipath and do not intend to ever use such hardware.
- You do not set disk.volumes = all.

> Note
> If you assign only certain drives to Swarm, disable multipath and only use a single channel of the multipath chassis. Although you could enable multipath just for those volumes, it would be hard to predict the resulting volume names for each drive.

Troubleshooting Multipath

Volumes are missing

If there are missing volumes (or no volumes) mounted from a multipath chassis, it might result from a disk controller problem. Look for a disk controller in the multipath chassis that is behaving incorrectly.

Volumes cannot be physically located

Check your configuration settings and make sure that you have not enabled multipath yet explicitly assigned specific volumes, which causes confusion: Because Swarm assigns multipath volume names non-deterministically, you have no way to know which multipath volumes will correspond to particular physical drives.

Contact Support for help finding the multipath device name for a physical drive.

Volumes are misnamed

Unless multipath is disabled, all volume names (including non-multipath volumes) must appear with multipath naming, such as `/dev/dm-1, /dev/dm-2`. This may be alarming if you expected non-multipath volumes to retain non-multipath naming, such as `/dev/sda`.

> Tip
> Contact Support for help adjusting your processes so that they do not require physical volume names.

Platform Implementation

Implementing Swarm Platform Server entails hardware provisioning, software installation and configuration of the Platform server, and Swarm Storage licensing and configuration. Implementation involves all roles needed to deploy and manage Platform:

- Storage system administrators
- Network administrators
- Technical architects

> Administering Platform Server
> See Swarm Platform topics under Swarm Administration:
>
> - Platform Administration
> - Platform CLI Commands
> - Platform UI
> - Platform Troubleshooting

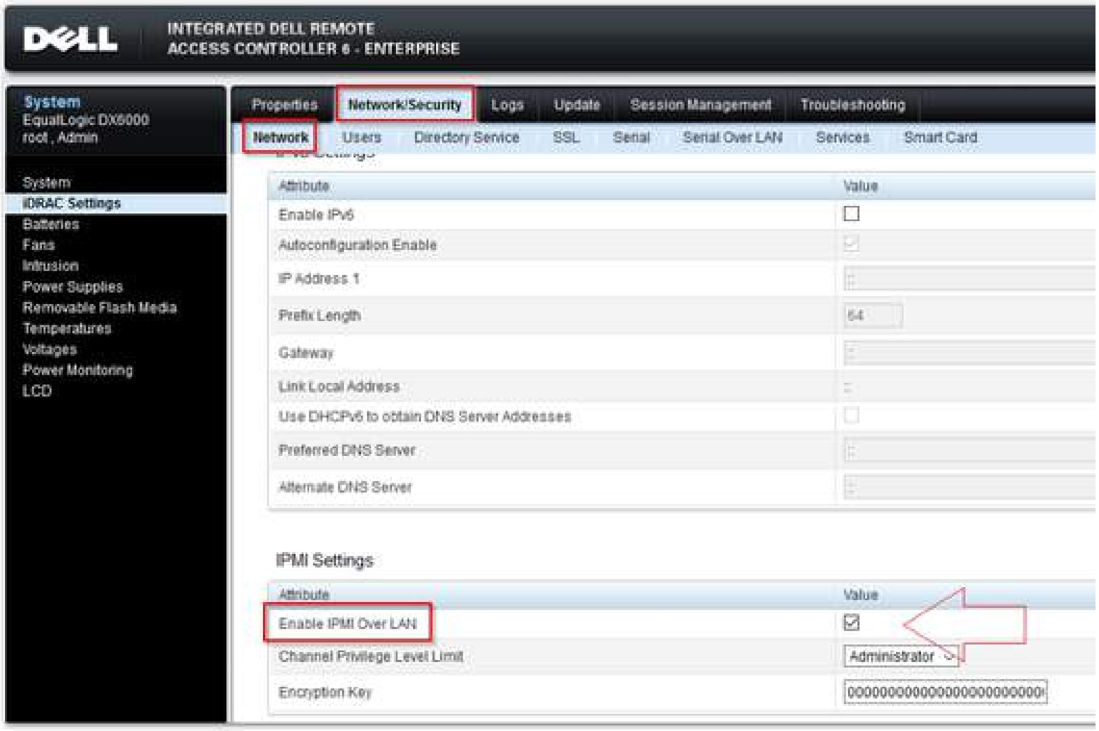
Platform Planning

- Platform Requirements
- Network Planning

Platform Requirements

Following are the requirements for the dedicated Platform server node that will manage your Swarm implementation:

System	2+ CPU cores 8+ GB RAM	Virtual machine The OVA file is configured to use 2 cores.
Storage	80+ GB available disk storage	Swarm nodes need to direct syslog logging messages at a dedicated partition on the Platform s Clusters that are large or have increased logging loads will require more storage for logging.

OS	Ubuntu Server 16.04 (Xenial)	http://releases.ubuntu.com/xenial/ or Platform Server OVA Image
Network	Static IP address	Before installing Swarm Platform, you must configure a network interface with a static IP addre network that will be managed.
IPMI	Enabled IPMI 2.0 or newer	To control the power state of managed chassis, Platform uses IPMI (Intelligent Platform Manag Interface), a standard for out-of-band (LOM, Lights-Out Management) device management and monitoring using a dedicated management channel. iDRAC-equipped nodes to be used for Swarm Storage must have the setting Enable IPMI Over LA led, which allows Platform Server to power the machine and commission the node.  DELL INTEGRATED DELL REMOTE ACCESS CONTROLLER 6 - ENTERPRISE System EqualLogic DX6000 root , Admin System iDRAC Settings Batteries Fans Intrusion Power Supplies Removable Flash Media Temperatures Voltages Power Monitoring LCD Properties Network/Security Logs Update Session Management Troubleshooting Network Users Directory Service SSL Serial Serial Over LAN Services Smart Card Attribute: Value Enable IPv6: ☐ Autoconfiguration Enable: ☑ IP Address 1: :: Prefix Length: 64 Gateway: :: Link Local Address: :: Use DHCPv6 to obtain DNS Server Addresses: ☐ Preferred DNS Server: :: Alternate DNS Server: :: IPMI Settings Attribute: Value Enable IPMI Over LAN: ☑ Channel Privilege Level Limit: Administrator Encryption Key: 0000000000000000000000000 Note If you cannot enable set iDRAC to Enable IPMI Over LAN, you will need to manually power chassis twice so that it can progress through the lifecycle. See Manual Power Control unc ploying Storage, below.

Network Planning

The Platform Server manages the allocation of IP addresses. There are three types of IP addresses (depending on the NIC configuration):

1. Dynamic Range — IP addresses used for enlistment and commissioning

> Requirements
> During bootstrap, Platform will reject a dynamic range that takes up more than 25% of the address space; if this is intended and necessary, you may add a `--force` flag to the bootstrap command.
>
> Platform will also reject any dynamic range that overlaps with IP addresses that are currently used.

2. Reserved Range — Special IP addresses for routers, network devices, etc.
3. Swarm nodes — IP addresses used for storage processes

Following are two scenarios for managed IP addresses:

Storage subnet	`10.0.1.0/24`	(255.255.255.0)
Set of all IP addresses	`10.0.1.1 - 10.0.1.254`	(254 addresses)
Platform Server's IP	`10.0.0.1`	(1 address)
Dynamic Range (via `platform bootstrap`)	`10.0.1.200 - 10.0.1.254`	(55 addresses)
Reserved Range (via `platform add iprange`)	`10.0.1.2 - 10.0.1.10`	(9 addresses)
IP addresses available for Storage nodes	`10.0.1.11 - 10.0.1.199`	(189 addresses)

Implications:

- There can be 189 Storage nodes on the network
- 55 chassis can go through enlistment/commissioning at a time

Storage subnet	`10.0.0.0/16`	(255.255.0.0)
Set of all IP addresses	`10.0.0.1 - 10.0.255.254`	(65,534 addresses)
Platform Server's IP	`10.0.0.1`	(1 address)
Dynamic Range (via `platform bootstrap`)	`10.0.0.11 - 10.0.0.110`	(100 addresses)
Reserved Range (via `platform add iprange`)	`10.0.0.2 - 10.0.0.10`	(9 addresses)
IP addresses available for Storage nodes	`10.0.0.111 - 10.0.255.254`	(65,424 addresses)

Implications:

- There can be 65,424 Storage nodes on the network
- 100 chassis can go through enlistment/commissioning at a time

Platform Installation

- Installation Checklist

- Install from OVA
- Install from ZIP
- Initialize the Platform Server

Installation Checklist

Before proceeding with the installation, gather the following information and files:

1. Determine the following:
   a. What subnet Platform will manage
   b. Which network interface on the server is connected to that subnet
2. Use the following Linux command to find the interface name:

   ```
   ifconfig
   ```

3. Document any IP addresses that should not be leased by Platform (reserved range).
   For example, for HSRP ranges: `x.x.x.1-x.x.x.3 or x.x.x.252-x.x.x.254`
4. Determine the number of nodes (chassis) anticipated for this cluster and select a subnet range for their IP addresses (dynamic range).
   For example, for a cluster of 10 chassis, the subnet range might be: `x.x.x.240 - x.x.x.249`
5. From Caringo Connect, download the latest bundle: `Swarm-v9.#-{date}.zip`
6. Transfer the Swarm components (ZIP packages) to the Platform server:
   a. Platform Server (non-OVA install) — `caringo-platform-{version}.zip`
   b. Service Proxy (to host the Swarm Storage UI, if Content Gateway won't be available) — `serviceproxy-{version}.zip`
   c. Swarm Storage — `storage-{version}.zip`

Install from OVA

1. Import the OVA file into your ESXi/vSphere environment. During the import process, add any network interfaces and configure as necessary.

   > Private subnet
   >
   > To support a private subnet for Storage and Elasticsearch nodes, the Platform server needs both public and private network interfaces (like the dual-network CSN).
   >
   > Important: With nodes on a private subnet, you must complete Deploying a Proxy to provide access to the health report server and other network resources, such as replication feed targets, NTP servers, and DNS. Without an interface into that private network, the Swarm UI cannot be accessed by an external browser.

2. Once the OVA is imported and booted for the first time, configure the networking as necessary within the OS.
   You can ssh into the Platform server using the following username/password: `caringoadmin/caringo`
3. Once the networking configuration is done and validated, run the following command where <static-ip> is the

static IP of the NIC connected to the storage network:

```
sudo /opt/reinit.sh -i <static-ip>
```

> Important
> If you enter the wrong IP address in the above command, the bootstrap command will be caught waiting and not return. See "Bootstrap Command Hangs" in Platform Troubleshooting.

4. When the `reinit.sh` script has finished, reboot the Platform Server.

Install from ZIP

> Required
> This process requires that you already have a running Ubuntu system with all the networking configuration done and working. If you plan to use a private network for your Swarm storage cluster, make sure that you have two separate network interfaces on your Platform server: one for the private subnet and one for the publicly available subnet.

1. SSH into the Platform server.
2. Unzip the package into an empty non-temporary directory and change directories into it:

```
unzip caringo-platform-{version}.zip
cd caringo-platform-{version}
```

3. Run the self-extracting installation script as a user with root privileges. For the following commands, use the interface name (such as `ens192`) that you found in step 2 of the Installation Checklist, above.
   For physical hardware or within a container, run:

```
sudo ./installplatform.sh -i {interface-name}
```

   For VMware ESXi virtual machines, run:

```
sudo ./installplatform.sh -i {interface-name} -e
```

   For additional command help, run the script without arguments: `./installplatform.sh`

Once installation completes, the CLI binary is installed, and it can be run from anywhere.

Initialize the Platform Server

> Required
> The bootstrap process requires Internet access.

1. Verify that there is at least 300 MB space available to run the bootstrap command, which will download updates from the Internet.
2. Once you have the interface name, you can view the list of available CIDR network addresses:

```
platform list subnets --all
```

3. Run the bootstrap command for the Swarm Platform environment, which includes preparing Swarm Storage for deployment to managed nodes.

```
platform bootstrap -i {Full Path to Swarm Storage ZIP} -v {#.#.#
Version of Swarm Storage}
 -t {DHCP Subnet CIDR calculated above} -a {First IP of Dynamic
Range} -z {Last IP of Dynamic Range}
```

- Storage — The Swarm Storage ZIP file for this command is located in the \Storage folder within the `Swarm-{version}-{date}.zip` download. The filename has this form: `storage-{version}-x86_64.zip`. You may change the version of Swarm Storage at any point after the bootstrap completes.
- Subnet — The “platform list subnets” command will give you the proper format. Example:

```
$ platform list subnets --all
Subnet: 10.1.1.0/24
Subnet: 172.17.0.0/16
Subnet: 192.168.68.0/24
```

  If the storage subnet is 192.168.68.x, then use 192.168.68.0/24
- Dynamic Range — You determined the Dynamic Range IPs in step 4 of the Installation Checklist, above.

> Caution
> The Dynamic Range specified in the bootstrap command must contain at least 10 IP addresses and also cover less than 25% of the total address space for the subnet. Taking up more than 25% of the addresses could result in too few IP addresses to boot Storage chassis. You can use a `--force` flag to force use of a range that violates these rules, but do so with caution.

4. If you have other IP addresses on the same network that will not be managed by Swarm Platform, create a reserved IP range so that their IP addresses are neither allocated by DHCP or used as IPs for Storage chassis:

```
platform add iprange -a {First IP of Reserved Range} -z {Last IP of
Reserved Range}
```

> Note
> Although you can run these commands at any time, doing so after chassis have been deployed means that you won't be able to reserve IPs that have already been allocated to Storage chassis.
>
> If you need to reserve IPs that have already been allocated to Storage chassis, you must release and redeploy the chassis. See instructions for Temporary Removal of a Chassis in Platform Administration.

5. Before deploying Swarm, upload a valid license in order for Swarm to boot successfully.

```
platform upload license -f {Path to license txt file}
```

Platform Settings

- NTP
- Syslog
- Networking

Swarm Storage nodes use a configuration file (.cfg). The file needs these minimum settings in order to boot Swarm:

```
cluster.name={Cluster Name}
disk.volumes=all
cluster.enforceTenancy=true
```

The Platform bundle includes a basic Swarm configuration file that it will auto-populate and apply to each new Swarm node under its management. Following are the network and management services that the Platform server handles for Swarm and the corresponding Swarm settings through which they are configured.

Make sure that the cluster.cfg is ready for deployment by the Platform server:

1. Edit the `cluster.cfg` file (`/etc/caringo/storage/cluster.cfg`)
2. Create entries for settings that Platform does not add (such as the proxyUri).
3. Create entries for any overrides that you want to make (such as to use a different time server).

NTP

The Platform server contains an NTP server and will automatically populate the following Swarm configuration setting to point to the Platform server, unless you override it with your own entry.

```
network.timeSource={Platform Server IP}
```

> Important
>
> The NTP server is essential because Swarm nodes must have a shared concept of time, and they must use the same time source, for consistency. If NTP is configured but unreachable, Swarm nodes will not boot.

Syslog

The Platform server contains a Syslog server and will automatically populate the following Swarm configuration setting to point to the Platform server, unless you override it with your own entry.

```
log.host={Platform Server IP}
```

Networking

The Platform server will automatically configure IP Address information for each chassis based on the IP ranges provided during installation. As such, the Platform server will automatically populate the following Swarm configuration settings, unless you override them with your own entries.

```
network.ipAddress={Required IP Address}
network.gateway={Platform Server IP}
network.netmask={Netmask for the Subnet}
```

Deploying Storage

To install Swarm Storage, you first enlist the chassis and then deploy them for management and orchestration. Swarm Platform supports any chassis that is equipped with IPMI 2.0 or newer.

1. Power on each chassis that will be a Swarm storage node. Each chassis will PXE boot from the Platform server, enlist itself with Platform, and shut itself down.
2. After enlistment completes, list the chassis that are available currently to run Swarm:

   ```
   platform list nodes
   ```

   The number of chassis returned will be the value to use in the next step for `-n`, unless you will be keeping some in reserve.
3. Deploy Swarm Storage to the chassis being managed by Platform:

```
platform deploy storage -v {#.#.# Version of Swarm Storage} -c
{Path to cluster.cfg} -n {# Nodes To Deploy}
```

4. At this point, the Platform will power on each of the nodes twice. The first power on is commissioning, involving hardware interrogation and other tasks, which ends with a power off. The second power on is to boot Swarm Storage on the chassis.
5. Once Swarm is running, deploy Content Gateway, or, if not using Content Gateway, proceed to Deploying a Proxy.

```
platform deploy proxy -b {Path to Service Proxy Bundle zip}
```

- Manual Power Control
- Network Bonding Mode
- Configuring Subclusters

Manual Power Control

When a chassis is powered on for the first time, the Platform Server auto-detects which power mechanism to use when processing them for deployment. However, if a chassis has IPMI for power control of the chassis but you need Platform server to ignore IPMI, you must switch the deployment to manual power control (add the `--manual` flag to the `platform deploy` command) and then manually power on machines at the right time as they go through the lifecycle.

When using manual power control, you need to use CLI commands to know when to power on each chassis, so use the `-y <system-id>` flag, rather than `-n <# of nodes>`.

1. Run the deploy command for an individual chassis, using the `--manual` flag and the `-y <system-id>` flag.
2. To know when to turn the chassis on, use the `--state Commissioning` flag:

```
platform list nodes --state Commissioning
```

3. As soon as the chassis appears in the list, you can power on the chassis.
4. After you power on the chassis, it will go through commissioning and then power itself off.
5. To know when to power on the chassis for the final time (to deploy the Storage software), use the `--state Deploying` flag:

```
platform list nodes --state Deploying
```

6. As soon as the chassis appears in the list, you can power on the chassis for the final time.

Tip
You can also use the following CLI command to follow the chassis through its lifecycle stages.

```
platform list nodes -y <system-id> --short
```

Network Bonding Mode

By default, the Platform server will boot all Storage chassis using the "balance-alb" bonding mode. You can modify the default bonding mode to use for future deployments by using the following command:

```
platform add bonding-mode --default-bonding-mode "<bonding-mode-to-use>"
```

Note
This setting is not retroactive. Changing the default bonding mode only applies to the chassis that are deployed after the default mode is set.

These are the available modes for the Linux bonding driver:

active-backup	Active-backup
balance-alb	(Platform default) Adaptive load balancing
balance-rr	Round-robin
balance-tlb	Adaptive transmit load balancing
balance-xor	XOR
broadcast	Broadcast
802.3ad	IEEE 802.3ad

Note
See Network Devices and Priority for details on bonding modes.

To use a different bonding mode when deploying an individual chassis, override the default value by using the `--bondingMode` flag:

```
platform deploy storage -v <#.#.# Version of Swarm Storage> -c <Path to
cluster.cfg> -y <system-id> --bondingMode "<bonding-mode-to-use>"
```

To modify the bonding mode for a chassis that's already deployed, run the `add kernelparam` command:

```
platform add kernelparam -y <system-id> --kernel_params
"castor_net=<bonding_mode>"
```

After the command completes, restart the chassis.

Configuring Subclusters

After all the chassis have been deployed and are running, you can assign chassis to subclusters.
To assign a chassis to a subcluster, use the `assign` command:

Add to subcluster

```
platform subcluster assign -y <system-id> --subcluster <subcluster-name>
```

> Note
> Assignment is not immediate. Allow time for every node on the chassis to be migrated to the new subcluster.

To review the subcluster assignments, use the `list` command:

List subclusters

```
platform subcluster list
```

Deploying a Proxy

To support a private subnet for Storage and Elasticsearch nodes, the Platform server needs both public and private network interfaces (like the dual-network CSN). With nodes on a private subnet, you must deploy a proxy to provide access to the health report server and other network resources, such as replication feed targets, NTP servers, and DNS. Without an interface into that private network, the Swarm UI cannot be accessed by an external browser.

- Deploying Service Proxy
- Deploying SCSP Proxy
    - Controlling the SCSP Proxy

Deploying Service Proxy

If you are not implementing Content Gateway, you need install Service Proxy on your Platform server to support and host your Swarm UI.

1. Locate the `Platform` folder in the Swarm 9 bundle ZIP, which you download from Caringo Connect.
2. Transfer the Service Proxy (`serviceproxy-{version}.zip`) file to the Platform server.
3. Run the following CLI command:

   ```
   platform deploy proxy -b {Path to serviceproxy.zip}
   ```

   The Platform server deploys the Service Proxy locally and completes all of the configuration needed to run it.
4. Query the status of the Service Proxy with the following CLI command:

   ```
   platform status proxy
   ```

5. Locate the logs for the Service Proxy at this location on the Platform server:

   ```
   /var/log/gateway
   ```

6. Log in to the Swarm UI. You can browse to the Platform server and it will redirect you to the UI: `http://{Platform-IP}/_admin/storage`

   ```
   http://{Platform-IP}
   ```

> Important
>
> If you later modify the password for the admin user or modify the membership of the storage cluster by adding or removing nodes/chassis, restart the Service Proxy so that it will register the changes:
> `platform restart proxy`

Deploying SCSP Proxy

If you need to deploy the SCSP Proxy to the Platform server, follow this process:

1. Download and transfer the SCSP Proxy to the Platform server.

2. Locate the SCSP Proxy (`scspproxy-8.2.1.zip`) file that you transferred to the Platform server.
3. Unzip the file and change directory into the folder named `scspproxy-8.2.1`
4. Run the following command:

   ```
   sudo apt-get install ./scspproxy.deb
   ```

5. Once the installation is done, modify the configuration files as appropriate:
   - `/etc/caringo/scspproxy/hosts.cfg`
   - `/etc/caringo/scspproxy/scspproxy.cfg`

Controlling the SCSP Proxy

Once the configuration files are modified as necessary, you can start and stop the SCSP Proxy using the following commands:

```
sudo systemctl start scspproxy
sudo systemctl stop scspproxy
```

Removing SCSP Proxy

To uninstall the SCSP Proxy from the Platform server, run the following command

```
sudo apt-get purge scspproxy
```

External Network Access

By default, the Platform server will configure all Storage chassis to use the Platform server as their default gateway. If the Storage chassis are on a private network and need to access resources that are outside the private network using a NAT service, you can enable this service on the Platform server as needed.

To allow the Platform Server to provide NAT for a private storage network, do the following:

1. Run the following command:

   ```
   sudo add_nat -e <NIC to external network> -i <NIC to private
   storage network>

   sudo add_nat -e eth0 -i eth1
   ```

2. After the command completes, reboot the server.

Storage Implementation

This section covers Swarm infrastructure and implementation:

- Network: Set up the Swarm network: it provides examples of a network configuration and describes how to set up your Swarm network infrastructure, including the network services, system booting, and the network devices.
- Hardware: Select system and storage hardware, with requirements and recommendations on how to set up the hardware devices in your storage cluster
- Installation: Install and configure a basic storage cluster, including information about licensing, booting your Swarm nodes, using the System Menu, and using Syslog.

> Tip
>
> Take advantage of the technical training videos available on Caringo Connect, where you download Swarm. Training video topics include Overview, Hardware, Configuration, Network Overview, Multiserver Config, Network Booting, Administration, and Swarm searching.

> See also these sections:
>
> - Swarm Storage Release Notes
> - Swarm Storage Cluster (configuration and administration)
> - Storage SCSP Development

- Implementation Checklist
- Licensing Swarm
- Installing and Initializing Swarm Storage
- Configuring Swarm for Gateway
- Drive Identification Plugin
- Swarm Passwords

Implementation Checklist

Here is a top-level view of typical issues to resolve before launching a production implementation of a Swarm cluster. These topics are covered in the sections that follow.

Hardware	• Swarm nodes meet minimum requirements: ◦ CPU: Intel or AMD, 64-bit, 2+ cores ◦ RAM: Sufficient memory available for anticipated object count ◦ Drives: SATA, SAS, or SCSI – enterprise-class with 24x7 duty cycle • Network: GigE or faster – multiple interfaces if redundancy required • Number of nodes is greater than the number of configured replicas and EC `k+p` • Sufficient free space in the cluster to recover content if a node fails?

Booting Type	• Platform Server, or • Self-boot options: • USB booting? Make sure configuration files are the same. • PXE/network booting? • Centralized configuration files?
Network Management	• DHCP or static IP addressing? • NTP server is configured and available. • Syslog server is configured and available; configured local6* and log rotation policy. • Network filters are configured to ensure only authorized users can reach the cluster. • DNS names for nodes are set up. • Network switch redundancy is sufficient for your deployment. • Monitoring procedures are in place: SNMP or assigned administrator watching console. • Verified bandwidth to cluster nodes. • Defined Swarm bonding mode.
Storage Config	• The `administrator` and `operator` default passwords have been updated to a new, secure value. • The node and/or cluster configuration file is updated and validated.
Data Protection	• Does the default number of replicas (`policy.replicas default`) meet your protection requirements? • Does the default EC ratio (`policy.ecEncoding`) meet your protection requirements? • Set lifepoints 'reps' for necessary file types? • File retention period and deleteability (yes or no) defined in lifepoints? • Any non-deletable files or automatic delete lifepoints? • UUID management mechanism protected (database or other means) if not using metadata search?
Default Domain	• Set `cluster.name` to your desired default domain name, a valid IANA FQDN host name (`cluster1.cloud.example.com)` • Create a domain that matches the value of `cluster.name`. This is the default domain for this cluster. • Set `cluster.enforceTenancy = True` to ensure that unnamed content is homed in a default domain if no domain is specified

Application Setup	● What storage protocols do your client applications use? ● If using HTTPS (SSL/TLS), create appropriate X.509 certificates and install on encryption off-load hardware/software. ● SCSP clients: ● Make sure relevant custom metadata is defined in natively integrated client. ● How does your application access the primary access node (PAN)? ● Create user accounts and, optionally, authentication tokens. ● S3 clients: ● Create S3 authentication tokens for applications ● Method for directing SCSP and S3 requests to appropriate Gateway port?
Testing	● Conduct end-to-end testing of the exact hardware, network setup, and software version to be used in production to ensure there are no compatibility issues. ● Test functionality using all client protocols.
Maintenance	● What is your disk drive replacement strategy? ● What are your maintenance windows for software updates? ● Do you use staging or testing environments to pre-qualify updates?

Licensing Swarm

When you purchase Swarm, you receive a license based on the contractually agreed amount of storage space you purchased for your cluster or a license is generated from the published aggregate of multiple capacity keys from the CSN Console. The `license.url` parameter in your node or cluster configuration file indicates the location of your license file.

> Important
> Swarm licenses are based on raw usage, the total footprint on disk. Be aware that increasing the number of replicas or EC parity segments increases the footprint, which has the effect of decreasing the net space that your license supports.

To evaluate your licensing, Swarm does the following:

1. Swarm starts with the default license, which is 2 TB per cluster.
2. Swarm attempts to read the license file from the location specified by `license.url`.
   By default, the license file is `/caringo/license.txt` on the node's USB flash drive or in the configuration file on the web or FTP server.
3. If the file specified by `license.url` is valid, Swarm uses it; if it is invalid, the storage capacity is set to `0`.

> Note
> If Swarm fails to read the license specified by `license.url` (for example, the file is on a web server that is temporarily unavailable), Swarm uses the last valid license.

- Registering for Caringo Connect

Registering for Caringo Connect

When you register for a Caringo Connect account, you can access a full-featured 2 TB Swarm license to evaluate the software. The initial license allows you to use the software until it expires on a specific date indicated in the `expirationDate` setting of the license.

If you exceed 2 TB of storage space or the expiration date has passed, Swarm generates the following `507 Insufficient Storage to Fill Request` message:

```
Not enough licensed space in cluster.
```

After you purchase Swarm with a contractually agreed amount of storage space, the `expirationDate` setting value in your new license will not include an expiration date.

```
expirationDate = none
```

> See the Sample License.

Troubleshooting licenses

If you experience issues with your license, try the following:

- If the capacity of a node in your cluster is `0`, make sure that the license specified by `license.url` is a valid Swarm license. In particular, make sure that no one has manually edited the license. This will invalidate the license key.

  > Note
  > Swarm checks every 15 minutes for license updates. If your license was modified, the electronic signature in the license becomes invalid, causing Swarm to revert to the default license.

- If the capacity of a node is set to the default of 2 TB for an extended period of time, ensure that:
  - You have a valid Swarm license.
  - You set the `license.url` to the location of the license.
  - The location you specified is available to the cluster. If you set `license.url` to a location that became unavailable to the cluster soon after it was booted, Swarm uses the default 2 TB license because it was the last known valid license.

Changing the license.url setting

You can specify alternate file names and locations using the `license.url` option in the node or cluster (Platform Server) configuration file. DNS names for FTP and HTTP hosts are supported as long as the DNS server and domain information is set by DHCP or is in the node and/or cluster configuration files. Errors processing the `license.url` setting are visible on the Linux system console during boot-up, but they will not prevent a successful boot.

Example `license.url` configurations:

```
license.url=http://192.168.0.103/license.txt
license.url=ftp://myftpserver/storagecluster_license.txt
license.url=file:///caringo/customerlicense.txt
```

> Important
>
> If you change the name of the license file for the local USB drive, the file must still remain in the `caringo` directory or one of its subdirectories.

Cluster Capacity Monitoring

> Important
>
> The maximum storage space you have in your cluster is the lesser of the two: the total physical space and the licensed capacity.

> Note
>
> The `featureMinReps` item in the license file is unrelated to the number of replicas required to be kept in the cluster (which you specify in your configuration), so it does not affect licensed capacity.

Running low on space — Two settings help you stay ahead of running out of cluster capacity:

- `console.spaceWarnLevel` specifies what percentage of space remaining on the node should trigger a warning. By default, it is set to 25% of available space.
- `console.spaceErrorLevel` specifies what percentage of space remaining on the node should trigger an alert. By default, it is set at 10% of available space.

Within an hour of reaching either of these thresholds, an announcement appears both in the logs and the UI.

Out of space — When available space in the cluster reaches zero, the immediate effect is that all inbound write requests are refused. However, internal replication and relocation continue, and all previously stored data is still readable. No data will be lost or corrupted.

> To increase your licensed capacity, contact your Sales or Support representative.

Upgrading a License

When you add additional storage capacity, a new feature, or updated customer information, you may be required to update the Swarm license file.

All new license files must be issued by your Sales representative (or generated from the Licensing page in the CSN

Console) to ensure that they have an electronic signature that is recognized and approved by Swarm.

> Important
>
> When updating the license file on a running cluster, be careful to match the path and filename that is defined in the `license.url` setting in the configuration.

How to update the license

The default location is the caringo directory of the local node's USB drive or centralized configuration server.

- If you are running Platform Server, upload the license using either the Swarm UI (Settings > License) or the CLI `platform upload license` command.
- If you are running CSN 8.3, upload the license using the CSN Console.

How Swarm validates the license

Swarm checks the license file every 15 minutes.

- If the file is updated, Swarm validates the license and updates the customer information and/or licensed capacity as necessary. On success, an announcement appears in the syslog and the UI.
- If a license update fails to validate a new license file, the cluster nodes will report a critical error in the syslog and the UI.
- If an update fails with a validation error, Swarm will continue to use the previous license file until the validation error is corrected.
- If the license file is located on an HTTP server that is not available when Swarm starts, Swarm analyzes the file for updates when it becomes available. Swarm may publish a corresponding update announcement in the syslog and the UI, even if the file itself has not changed.

Sample License

Below is a sample of a Swarm license.

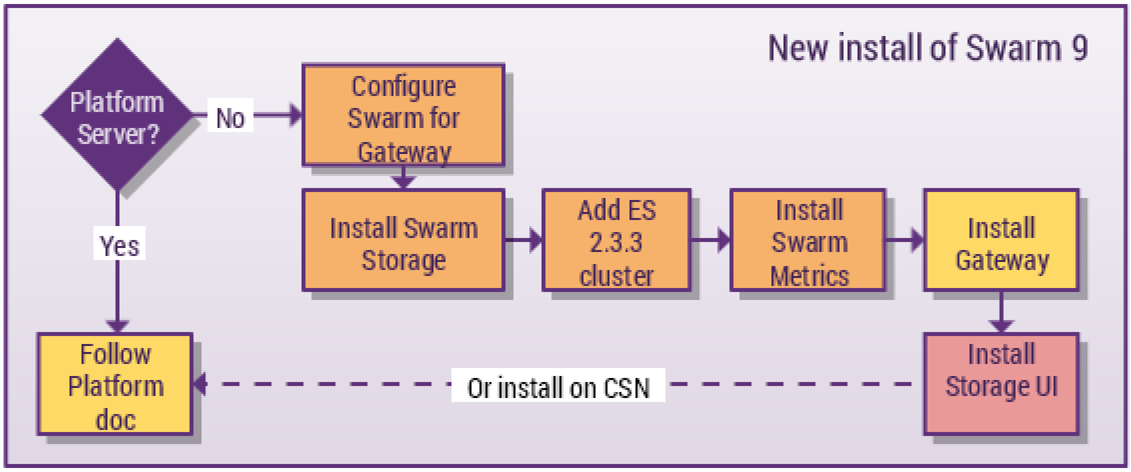
```
----BEGIN PGP SIGNED MESSAGE----
Hash: SHA1
#
########################################################################
###
# Swarm License File
# License S/N: 200804261512-8402
# Generated By: Eric Smith
# Comments:
#
########################################################################
####
licenseFormat = 1.1
cn = ACME Widgets, Inc.
street = 123 Street A, Building #23
cl = Austin
st = Texas
postalCode = 78746
co = USA
clusterDescription = Corporate Office
# License Components
expirationDate = none
featureClusterMaxTB = 2
featureContentIndexing = no
featureErasureCoding = yes
featureMinimumMinReps = 1
----BEGIN PGP SIGNATURE---
Version: GnuPG v1.4.6 (GNU/Linux)
iD8DBQFIHztdRYikRJU1RfMRAusHAKCX9ABhEBgQz/TyTy+gT5gXf7hNmQCeKxLx
R+AgQ2uoR/l+mG4Apx6zgDk==TXHc
----END PGP SIGNATURE----
```

Installing and Initializing Swarm Storage

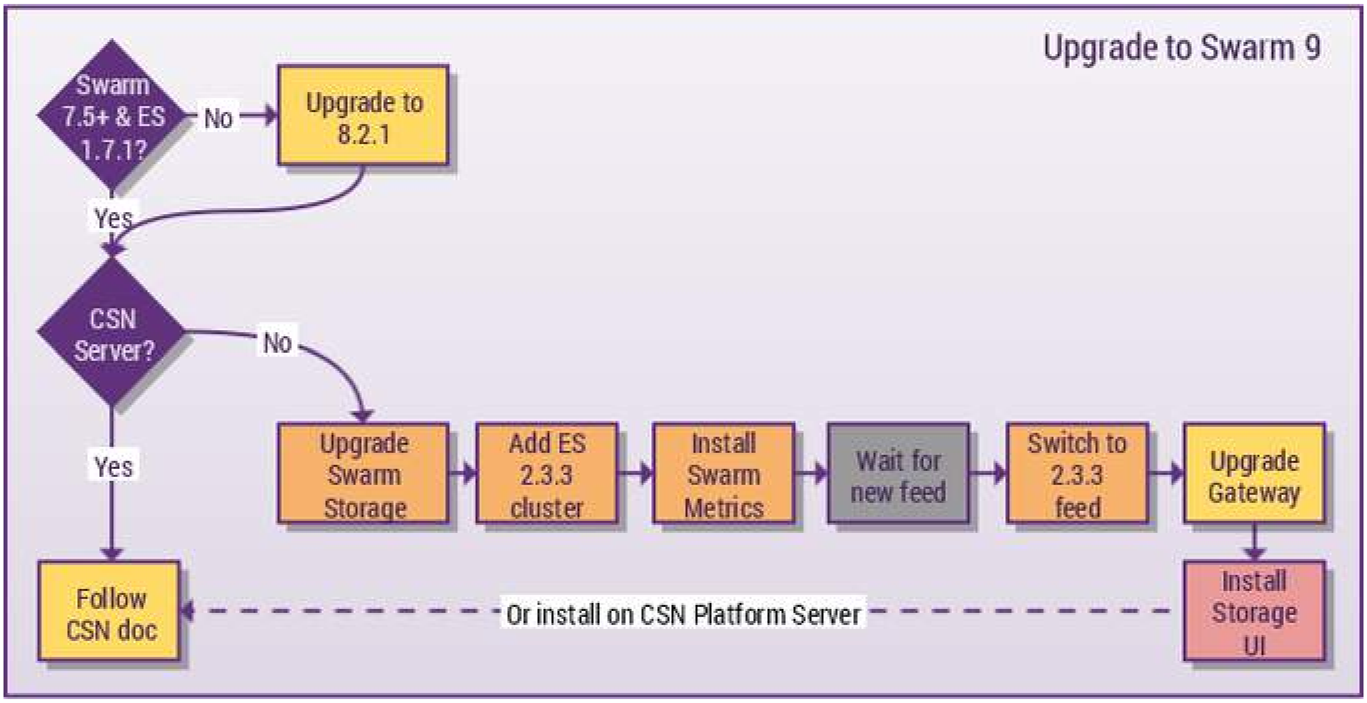
This section describes how to set up and configure a basic storage cluster.

> Platform Server
>
> If you use Platform Server, skip this section: your cluster is set up.

This is an overview of the process for installing Swarm 9:



- Configuring Swarm for Gateway
- Initializing a Storage Cluster
- Installing Elasticsearch
- Installing Swarm Metrics
- Gateway Installation
- Swarm Storage UI Installation



- CSN Installation
- Upgrading a Storage Cluster
- Upgrading from Elasticsearch 1.7.1
- Installing Swarm Metrics
- Upgrading the Gateway

- Swarm Storage UI Installation

Initializing a Storage Cluster

This section describes how to initialize a Swarm node.

- Editing the Configuration File
- Configuring the Storage Volumes
- Configuring the DHCP or Static Network
- Setting the Admin Passwords
- Booting Swarm Nodes

> Platform Server
> If you use Platform Server, skip this section: these tasks are performed for you.

To set up a new USB flash drive with Swarm, follow the readme file in the software distribution that describes how to set up the drive and what to copy over from the software distribution. You must complete those steps before you start editing node.cfg on the USB drive.

> Warning
> Make sure that no one (including IT personnel and any on-site contractors) reboots a non-Swarm system while a Swarm-configured USB is mounted. If a computer accidentally boots from a Swarm USB, its drives could be reformatted, with permanent data loss.

Editing the Configuration File

This section describes how to configure a storage cluster node by manually editing the node and/or cluster configuration file.

> Note
> A failsafe timer mechanism is included in the Swarm startup process that restarts the boot process if the boot error screen displays for more than 15 minutes. This reboot is intended to compensate for temporary network loss and is canceled if you use a keyboard on the console.

To manually edit the node.cfg file:

1. In a text editor, open the .cfg file on the USB flash drive or PXE configuration server.
2. Set the setting disk.volumes = all. This setting lets Swarm use all available volumes on the node.
3. Set the cluster.name configuration setting to the name of your cluster. Use an IANA-compatible domain name, such as `cluster.example.com`.

   > Important
   > Configure all nodes in the cluster with the same cluster name.

4. If you are not using DHCP, you can modify the network configuration to assign node IP addresses.

5. Set other configuration settings as desired.
6. Save and close the .cfg file.
7. If your node boots from a USB flash drive, make sure to safely unmount or stop the USB flash drive; otherwise, your changes will not be saved to the .cfg file.
8. Make sure a valid license.txt file is located in the caringo directory.
9. Set up and configure the NTP server. (See Configuring an External Time Server.)
10. Set up and configure the syslog server. (See Configuring an Rsyslog Server.)
11. Boot the Swarm node.

> Important
> Multiple reboots in a row indicate there might be a problem with the system configuration. If this occurs, check the syslog server for errors.

Configuring the Storage Volumes

Swarm reads the disk.volumes setting from the node and/or cluster configuration files to determine which disks can be used for content storage. Swarm is pre-configured with a dummy value that prevents the disks on your cluster nodes from being reformatted during the installation procedure. After you install Swarm, an administrator must edit the disk.volumes setting value with the keyword all or with a list of disks that Swarm can use for storage.

The easiest way to use all disks in a node for content storage is to use the keyword all:

```
disk.volumes = all
```

When using the all keyword, Swarm automatically excludes the Swarm USB flash drive from being used for storage.

> Important
> Hot plugging is only supported for disk.volumes = all.

To set the disk device names, set disk.volumes to a space-separated list of drive identifiers.

Swarm uses standard Linux volume identifiers such as/dev/sda and /dev/sdb. If you cannot identify your volume identifiers, you can use the Swarm System Menu or you can access the Swarm node using another Linux system.

Using the preceding example, set disk.volumes to:

```
disk.volumes = /dev/sda /dev/sdb
```

> Warning
> Make sure that no one (including IT personnel and any on-site contractors) reboots a non-Swarm system while a Swarm-configured USB is mounted. If a computer accidentally boots from a Swarm USB, its drives could be reformatted, with permanent data loss.

Configuring the DHCP or Static Network

The easiest way to set up your cluster is to configure DHCP to automatically assign an IP address to each cluster node. DHCP is used by default in a CSN configuration. Otherwise, you need a DHCP server in the network that is connected to your cluster nodes. When completed, Swarm can be booted from the USB flash drive or network environment.

To assign static IP addresses to your nodes, edit the network.ipAddress, network.netmask, and network.gateway settings in the node and/or centralized configuration files. Set all three settings when using the static IP address configuration option. In a centralized configuration environment, these settings must be set in a custom configuration file on each cluster node because this information cannot be shared between the nodes.

To edit the IP address, network mask, and default gateway settings on your nodes:

1. Open the centralized configuration file, or open the configuration file on the first node.
2. In the file, edit the network settings.

   Network settings

   ```
   network.ipAddress = 10.20.30.101
   network.netmask = 255.255.255.0
   network.gateway = 10.20.30.1
   ```

3. If you are editing the node configuration file, repeat for each node. If you are editing the centralized configuration file, you are done.

Setting the Admin Passwords

The Swarm `admin` password is required to make changes in the Swarm Storage UI; read-only credentials (`operators`) are needed to view it. See Defining Administrators and Users.

> Important
> When you deploy your storage cluster, be sure to change the default passwords.

As of Swarm 10.0, there are two pairs of passwords to manage, which are in the cluster-wide `[security]` and `[snmp]` sections of the Swarm Storage configuration file (`cluster.cfg` (Platform/CSN) or else `node.cfg):`

Setting name	Default	Descriptions and Examples

security.administrators	`{'admin': 'ourpwdofchoicehere'}`	One or more username:password pairs. Sets credentials for who can administer the cluster via the Swarm UI. Upgrading from 9.x If the value includes the snmp username, remove it from here and update `snmp.rwCommunity` with its password. Example: `{'admin': 'adminpassword','admin2':'adminpassword2'}`
security.operators	`{}`	One or more username:password pairs. Sets credentials for who can view the Swarm UI. Upgrading from 9.x If the value includes an snmp username, it is ignored; remove it from here and update `snmp.roCommunity` with its password. Example: `{'operator': 'operatorpassword','operator2': 'operatorpassword2'}`
snmp.rwCommunity	`ourpwdofchoicehere`	String. Password for the SNMP read-write community. Important You must know the SNMP read-write password in order to dynamically change the Swarm '`admin`' password. To change the SNMP read-write password, you must edit the config file.
snmp.roCommunity	`public`	String. Password for the SNMP read-only community.

Caution

- The name **`admin`** is reserved, so do not delete it, which could cause errors and affect performance. If you decide not to use **`admin`**, define a complex password to protect it.
- Swarm prevents cluster booting if the SNMP security administrator (read/write user) is not set properly in the configuration file.
- All administrative users and passwords must agree on all nodes or certain cluster actions will fail.
- Password updates are not complete until they are persisted in the cluster settings file across all nodes, and rapid, successive updates cannot be accepted on a given node until the first update completes processing.

Encrypting Passwords

Instead of a clear text password, you can represent the password as a hexadecimal-encoded MD5 hash of the following string:

```
username:user-list-name:password
```

where username and password consist only of ASCII characters and `user-list-name` can be either "CAStor administrator" or "CAStor operator".

To create the MD5 hash, use a programming language or a utility such as md5sum or Apache htdigest. For example, to update your node or cluster configuration file with a password hash you create using htdigest:

1. Create a file that contains a hash of the user name, password, and user list name:

   ```
   htdigest -c password-file.txt "CAStor administrator" Jo.Jones
   ```

2. When prompted by htdigest, enter and confirm the user's password.
3. Open the new file (`password-file.txt`) in a text editor. The hash is the last entry in the string:

   ```
   Jo.Jones:CAStor administrator:08b0468c1d957b7bac24463dd2191a2d
   ```

Updating Passwords

How you update the passwords depends on which ones need updating and whether Swarm has ever been started.

> Note
> If you do not use a Platform Server, make these changes to the `node.cfg`.

Situation	Process	Examples and notes

<table>
<tr>
<td>Swarm has never booted</td>
<td>
<ol>
<li>Create and hash an <code>admin</code> password.</li>
<li>Update passwords in the config file.</li>
<li>Important: If booting from a USB flash drive, be sure to unmount/stop the USB drive or else the changes will not be saved.</li>
<li>Boot the Swarm cluster.</li>
<li>After the cluster is running, you can remove the password from the config file.</li>
</ol>
</td>
<td>
Hash of password
<pre>
$ echo -n 'admin:CAStor
administrator:NEWPASSWORD' | md5sum | cut -d
' ' -f1
7fe563b8532b3a460def0895895eebf5
</pre>
The first time that you boot the cluster, the Swarm admin password must be in the config file:
<pre>
[security]
administrators =
{'admin':'7fe563b8532b3a460def0895895eebf5'}
</pre>
When the cluster is running, Swarm stores the admin password in the persisted Settings object, at which point it is safe to remove the password from your configuration file for security purposes:
<pre>
[security]
administrators = {}
</pre>
</td>
</tr>
<tr>
<td>Updating SNMP passwords</td>
<td>
<ol>
<li>Update passwords in the config file.</li>
<li>Reboot the Swarm cluster.</li>
</ol>
</td>
<td>
Important
You must know the SNMP read-write password in order to dynamically change the Swarm '<code>admin</code>' password. If you need to change the SNMP read-write password, you must edit the config file.
After rebooting with the new SNMP password in the file, you can proceed to change the Swarm '<code>admin</code>' password.
</td>
</tr>
</table>

Updating Swarm admin password	1. Create and hash an `admin` password.
	2. Update password via SNMP, which Swarm will save in the persisted Settings object.
		Changing admin password
		`snmpset -v2c -c SNMP- password -m +CARINGO-CASTOR-MIB Swarm- node- IP addModifyAdministrator s "admin:new-password"`
		`snmpset -v2c -c ourpwdofchoicehere -m +CARINGO-CASTOR-MIB 172.20.3.85 addModifyAdministrator s "admin:7fe563b8532b3a460def0895895eebf5"`

Frequently asked questions:

- How do I change the active SNMP read-write password? The SNMP passwords cannot be changed dynamically. Changing one or both requires a config file update and a cluster reboot.
- What is the SNMP read-only password? The read-only password '`public`', which is the 'community string'
- Is the read-only SNMP password in the persisted Settings object? No
- Can my SNMP read-write passwords in the persisted Settings object and cluster.cfg be different? Yes, but only the config file SNMP read-write password is used.
- How do I change my admin password? Update the password via SNMP and then update it in the config file, unless you've removed it from there.
- How do I change my SNMP read-only password to the cluster? Change the `snmp.roCommunity` setting in the config file and reboot the cluster.

Booting Swarm Nodes

If you are booting your nodes from a USB flash drive, ensure that the basic configuration requirements are completed and the USB flash drive is inserted into the USB port on your cluster node. After the hardware self-test is completed, the Swarm operating system loads into system RAM and the caringo\node.cfg file is read from the USB flash drive.

If you are booting your nodes from a network, ensure that the network boot environment is configured and online. When completed, you can boot your nodes.

You may attach a monitor to the node, but it is not required.

Upgrading a Storage Cluster

This section describes how to upgrade your Swarm license and cluster nodes.

> Platform Server
> If you use a Platform Server, see Platform Administration for how to upgrade Swarm (for CSN 8.3, see CSN Updates and Upgrades).

- Types of upgrades

- Preparing for the upgrade
- Upgrading the nodes
- Example shutdown script

Types of upgrades

Unless specifically noted for a particular release, a single cluster can contain nodes running mixed versions during the upgrade process and no data conversion between versions is necessary.

- Simple upgrade – The simplest upgrade method is to reboot the entire cluster at once after the software on all USB flash drives or the centralized configuration location has been updated.
  1. Shut down all nodes in the cluster.
  2. Upgrade the software.
  3. Reboot the nodes.
  4. Verify that all nodes are healthy.
- Rolling upgrade – To upgrade your cluster without scheduling an outage or bringing down the cluster, restart your nodes one at a time with the new version and let cluster continue serving applications during the upgrade process. If you have stored your objects with at least two replicas, they will continue to be fully accessible during the upgrade. If you are using this rolling upgrade approach, wait at least 10 seconds between each node reboot to ensure that each node can properly communicate its rebooting state to the rest of the cluster and ensure other cluster nodes do not initiate recovery for the rebooting node.

> Errors
>
> - Ongoing processes — If there are any disk recovery or retire processes ongoing in the cluster during a rolling upgrade, you may see errors in your log similar to '`Castor-System-Cluster value must refer to a remote cluster on RETRIEVE request`' . These errors are harmless and will stop once all nodes are running the new version.
> - Blocked feeds — When you start a rolling upgrade from Swarm 8, Swarm 9 modifies the feed definition in the persisted Settings object. That change is not supported by Swarm 8, so those nodes get blocked feeds with a config error ("Plugin validation error: Unknown attribute indexAlias") and are unblocked only when the last Swarm node has been upgraded. During the rolling upgrade, the feed will be blocked on some nodes, which may not support indexing and querying. Should you ever downgrade to Swarm 8, the feed will block again in the same way: either delete the feed and redefine it, or contact Support for help updating the feed definition in the persisted Settings object.

Preparing for the upgrade

To prepare for the upgrade:

1. Download the upgraded Swarm software from Caringo Connect.
2. Important: Review the release notes for upgrade instructions that are specific to the version that you downloaded.
3. Important: Run the Storage Settings Checker and resolve any configuration issues with Caringo Support.
4. Locate your node configuration data, backup configuration files, and license files.
5. Prepare your node configuration data on new USB flash drives or on a centralized configuration server.

6. Verify the health of all cluster nodes.
7. Schedule an off-line window for the cluster down time.

Review the release notes included with the new boot devices prior to starting the cluster upgrade. The release notes contain information about feature changes, operational differences, and any issues that could affect how your storage cluster will process and store data.

You can remove USB flash drives from the running nodes to view and back up the configuration and license files. USB flash drives or the configuration server can be updated using the instructions in the `README.txt` file found in the latest Swarm installation package. After performing the upgrade, validate your node or cluster configuration file to make sure there are no deprecated parameters that need to be removed or renamed.

> See Configuring the Nodes for how to set the parameters in the node or cluster configuration files.

After all upgrades and validations are completed, you can return each USB flash drive to the node from which it was removed. Be sure to match each USB device to its original node in the cluster and make sure the `vols` parameter which defines the storage devices matches the correct node.

> Important
>
> Before you perform any node upgrade, verify the cluster health by checking for critical error messages on the status page of each node or the SNMP `CastorErrTable` OID. This process ensures that no hardware problems exist that could interrupt the upgrade process. Any problems should be corrected prior to upgrading your cluster.

When upgrading a single node in your cluster, be sure to include the `clusterSettingsUUID` parameter value in the node or cluster configuration file prior to rebooting the node so the settings file can be located after the nodes reboot.

Upgrading the nodes

To upgrade your cluster nodes:

1. Shut down all cluster nodes (or one at a time for a rolling upgrade).
2. Install the updated USB flash drives on your PXE boot server.
3. Reboot all nodes.
4. Verify that your cluster is operating normally.

A simultaneous shutdown of all cluster nodes is the first step in the simple upgrade. If your cluster cannot be shut down during normal business operations, the nodes can be rebooted one at a time in a rolling upgrade so the cluster remains online.

If you perform a rolling upgrade and a cluster node is off-line, the cluster will detect the missing node and the remaining nodes will try to recover the missing content via the failed volume recovery (FVR) and erasure coding recovery (ECR) processes. When the missing node is brought back online, the cluster detects the node and the recovery processes will stop.

> Tip
>
> To minimize node downtime and prevent the remaining nodes from filling with recovered content, prepare USB flash drives with the upgraded version of Swarm for each node before you begin the upgrade.

If all the nodes are shut down within several seconds of each other, initiating the disk recovery process is not a concern. You may also choose to suspend volume recovery from the Setting window on the Swarm Admin Console to prevent recovery from kicking off while nodes reboot into the new software version.

> See the Suspend Setting.

Example shutdown script

This UNIX shell script demonstrates a method of issuing the shutdown command to all cluster nodes. In this example, all the nodes of the cluster are defined in the NODES variable.

```
NODES="192.168.1.101 192.168.1.102 192.168.1.103"
for n in $NODES;
 do snmpset -v 2c -c pwd -m +CARINGO-CASTOR-MIB
  $n caringo.castor.CastorShutdownAction = "shutdown"
done
```

Storage Settings Checker

- Running the Checker (CSN)
- Running the Checker (Non-CSN)
- Using the Results

Swarm Storage versions 10 and higher are supported by a configuration checker, which is bundled with the support tools that you download from Caringo Support. The report will establish the current configuration across the cluster and will surface any setting issues (such as deprecations and new requirements) that need to be addressed. (v10.0)

To evaluate all of the active settings for your Swarm cluster, the checker works with both types of configuration settings:

- Dynamic — The persisted settings stream (PSS object) for your cluster, which is updated in real time by the Swarm UI or SNMP.
- Static — The configuration file(s) that are used at node startup.

> Best practice
> Before troubleshooting Storage or starting any Storage 10.x upgrade, run the latest settings checker and verify the results with Support.

Running the Checker (CSN)

For CSN sites only, you can use the techsupport-bundle-grab.sh with the -s option to run the settings checker and bundle the output along with the other needed logs and support information:

1. On the Support site, download the latest support bundle: caringo-support-tools.tgz

> Important
> When using this tool for settings analysis before upgrading, always be sure that you have downloaded the latest version available.

2. Expand the support tools and run the following script with the settings option, which will generate the complete tarball:

   ```
   techsupport-bundle-grab.sh -s
   ```

3. On the Support site, create a new ticket for your configuration review, and note the ticket number (such as SUP-1234), which you will need later.
4. Upload your tarball.
   - If the machine has access to the internet, run this command for direct upload:

     ```
     uploadtosupport [output-file]
     ```

   - If not, use `scp` to secure copy the file to a location with internet access and then use the Support Uploader.
5. When prompted, enter the ticket number. The tarball will be attached to it.
6. Support will be notified of the upload and will work with you on any configuration issues that are surfaced.

Running the Checker (Non-CSN)

For non-CSN sites, you will run the settings checker as a standalone Python script:

1. On the Support site, download the latest support bundle: caringo-support-tools.tgz
2. Expand the support tools, and run the following Python script with the needed options:

   ```
   python settings_checker.py [options]
   ```

   See Script Options below.
3. The script compresses the outputs into a tarball archive, which is the preferred format. Use `scp` to secure copy the file to a location with internet access.
4. On the Support site, create a new ticket for your configuration review, and note the ticket number (such as SUP-1234).
5. Upload your tarball with the Support Uploader.
6. When prompted, enter the ticket number. The tarball will be attached to it.
7. Support will be notified of the upload and will work with you on any configuration issues that are surfaced.

Script Options

Which options are required depends on what type of Swarm environment you are running:

- In CSN environments, none of these options are required if the configuration file in the default path has the correct username and password. The only flag typically needed in this environment is -w PASSWORD.
- In non-CSN environments, you must provide options (user, settings) that enable locating and loading of the dynamic settings (PSS).

Typical usage

- Add `-c` to see the output on the console.
- For a CSN site, add only the `-w PASSWORD` option, for the Swarm admin.
- For a non-CSN site,
  1. Locate the settings config file — Add either `-f FILE` (for one or more file paths) or `-p PATH` (for a set of config files).
  2. Locate the PSS (persisted settings stream) — Add the `-s SETTINGS` option, with -u USERNAME_PASSWORD for admin authorization.

Options	Rules	Scope	Notes
`-h, --help`			Show program help and exit.
`-v, --version`			Show program's version number and exit.
`-o OUTPUT`			The pathname of the output file, which opens in append mode. Defaults to timestamped output in the current working directory: `./swarm_settings_$HOSTNAME_v$TOOLVERSION_$TIMESTAMP.out`
`-c, --console`			Disabled by default. Print to the console, as well as the output file.
`-m`			Display more output.
`-w PASSWORD`	Do not use with *-u*	CSN	The password of the cluster admin, used to retrieve the cluster's PSS (persisted settings stream). If not specified, the password is read from the cluster.cfg file.
`-u USERNAME_PASSWORD`	Do not use with *-w*	Non-CSN	The `username:password` of the cluster admin, which is needed to read the PSS. If not specified, the username is read from the node.cfg file.
`-p PATH, --path=PATH`	Do not use with *-f*	Non-CSN	If not specified, the script defaults to legacy CSN configuration locations.
`-g SUFFIX`	Only use with *-p*	Non-CSN	Optionally, specifies the file type, if other than '`.cfg`'.
`-f FILE [FILE ...], --file FILE [FILE ...]`	Do not use with *-p*	Non-CSN	The location of the configuration file (or files separated by a space), or the path to all configuration files. Defaults to file type '`.cfg`'
`-s SETTINGS`	If not known, use *-a*	Non-CSN	If known, the host/UUID of the PSS for your cluster, in URI form: `http://HOST/UUID.`

`-a ADDRESS`	Only provide if requested	Non-CSN	The IP of a node in the cluster, to help obtain the PSS.
`-r PSSFILE`		Non-CSN	The location of a PSS file.

Using the Results

Review the generated output, which will comprise these reports, warnings, and checks, and take action accordingly:

Reports	• Reports the tool version. • Reports the Swarm version of the current settings file. • Reports the hostname. • Reports the location of the logging output. • Reports the discovered IPs. • Reports the known chassis count. • Reports all static IPs in `network.ipAddress`. • Reports cluster policy values. • Obscures all secure keys. • Allow a verbose mode to report when the default values are explicitly defined for settings automatically set to their defaults by CSN: `cip.group, cluster.settingsUuid, cluster.proxyIPAddress`. • For each config file in a CSN environment, show only the explicit overrides from that file (if allowed by --`verbose` mode).

Warnings	• Validation errors that would prevent a node from booting. • Special case settings we recommend changing, like passwords. • When defined settings have been deprecated, regardless of value. • With the PSS: ◦ a defined value does not match the saved value ◦ a defined value matches the saved value ◦ a default value does not match the saved value • Without the PSS: ◦ a defined value changed the default ◦ a defined value matches the default • When a setting is not in PSS: ◦ a defined value changed the default ◦ a defined value matches the default • When a setting will be newly saved to the PSS. • When `security.administrators` is in the config file. • When cluster-level settings are overridden in node configs. • When `index.ovMinNodes` is modified, and there are not enough nodes. • When a process count other than 1 is specified in proc_count.cfg or pc<mac>.cfg. • When a maxChassis value other than the default 16 is specified in netboot.cfg.
Checks	• Checks for `ec.protectionLevel` = subcluster, and only 1 value for `node.subcluster`. • Checks for when the node count < 3. • Checks for when the node count < `policy.replicas min` reps. • Checks for when there are too few "levels" for the EC policy setting, base on `ec.protectionLevel`. (Volume-level protection cannot be validated.) • Checks for when there are < p+1 nodes when EC is allowed by policy. • Checks for when `index.ovMinNodes` < node count. • Checks for when cluster-level settings are inconsistent across configs.

If you have any questions about resolving any of the issues that surface, contact Support before proceeding with your upgrade.

Rebooting the Storage Cluster

After the cluster is shut down, make sure the updated USB flash drives, configuration server, or CSN is prepared and then begin the reboot process.

This is the recommended power-on sequence:

1. Start a single node.
2. Verify that the node boots properly. This process verifies that the software loads correctly and the node can communicate with the network.
3. Repeat on each remaining node in the cluster.

When you initially boot a node, the `hpStartDelay` parameter provides a 15-minute delay window by default for all nodes to boot up and join the cluster before Swarm begins to check for missing cluster nodes. As long as all nodes in the cluster are running within this window, you can avoid failed volume recovery (FVR) and erasure-coded recovery (ECR) operations.

This window only exists immediately following a node reboot. After 15 minutes, the recovery operations begin immediately after Swarm detects a missing node.

Restoring a previous version

If you need to install a legacy software version on any node, the software can usually be reverted to a previous version using the same method performed during an upgrade.

Limitations

- Due to internal Swarm changes, reverting to a previous version is not supported when upgrading to version 6.0 or later from a version prior to 6.x.
- The 6.5.1 release can be downgraded to any 6.x release except 6.5.0.

Using the System Menu

- Configuring a Node with the Setup Wizard
- Formatting a Drive
- Reformatting all Drives in a Cluster

The System Menu provides you with general information about your system. From the System Menu, you can:

- Review system information
- Diagnose startup errors
- Format hard drives
- Perform diagnostics
- Shut down and restart the cluster nodes

> Caution
> Use the option to reformat all devices with care. Using the System Menu, you can reformat all discovered devices, including your Swarm USB flash drive. Make sure that you remove your USB flash drive before you choose this option.

Before you begin, make sure that you configured your node and/or cluster configuration file.

> See Editing the Configuration File.

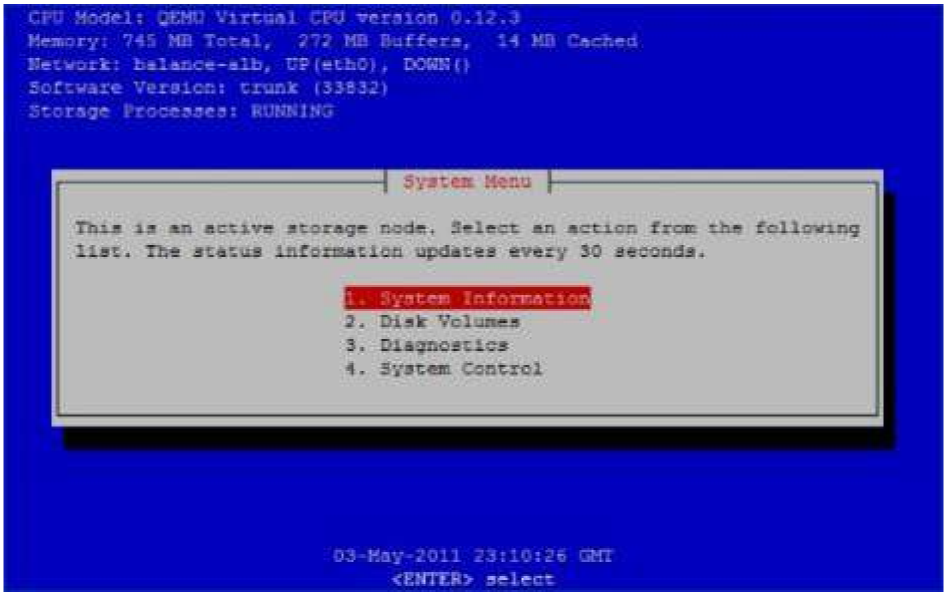
Menu	Usage
System Information	1. Configuration - Displays the current node configuration. Press : q to close. 2. Startup Errors - Displays the boot errors. This option displays only if errors appeared during boot-up.

Disk Volumes	Displays a list of all detected volumes of at least 8 GB in size. Select a volume and enter one of the following: • F to format the volume • I to identify the volume by flashing its LED • ALL to perform F and I on all drives Caution If you select ALL and press F, all discovered devices are reformatted, including your Swarm USB flash drive. Remove your USB flash drive before you choose this option.
Diagnostics	Assists you with gathering network and log information and provides utilities such as `ping` and `traceroute`. Select an option from the menu and follow the prompts on your screen to complete the tasks. 1. Kernel log (dmesg) 2. Socket connections 3. ARP cache 4. Routing table 5. Network Interface Details 6. Time synchronization (NTP) 7. Ping a host 8. Traceroute to a host 9. Hardware clock 10. Netstat send/receive buffers 11. Castor Startup log (v10.0)
System Control	Lets you manage the node: 1. Reboot System - Restart the node. 2. Shutdown System - Shut down the node (that is, stop all processes and power it off). 3. Stop Storage Processes - Stops the running Swarm processes so you can reformat volumes if needed.

Configuring a Node with the Setup Wizard

To use the setup wizard to configure a node:

1. Temporarily connect a keyboard and monitor to the node.
2. Boot the node. After the node boots, a screen similar to the following appears:



```
CPU Model: QEMU Virtual CPU version 0.12.3
Memory: 745 MB Total,  272 MB Buffers,  14 MB Cached
Network: balance-alb, UP(eth0), DOWN()
Software Version: trunk (33832)
Storage Processes: RUNNING

                    System Menu

  This is an active storage node. Select an action from the following
  list. The status information updates every 30 seconds.

                    1. System Information
                    2. Disk Volumes
                    3. Diagnostics
                    4. System Control

                 03-May-2011 23:10:26 GMT
                      <ENTER> select
```

Messages at the top of the screen display system information, which includes:

- CPU information, such as model and clock speed.
- Memory information, including the amount of buffer memory and cached memory.
- List of all network adapters and their related status (up or down).
- Swarm software version.
- Storage processes, such as Storage Processes: RUNNING that indicates the node booted successfully.

> Note
> If the node did not boot successfully, the Attention page displays. Select Enter, OK, System Information to view the errors that prevented the node from booting.

3. Select an option from the System Menu:
   - System Information
   - Disk Volumes
   - Diagnostics
   - System Control

Formatting a Drive

If a drive has never been seen by Swarm and it boots into a Swarm chassis, the drive will automatically format on bootup (or insertion, in the case of hot plug). However, if a drive has been retired from a cluster and needs to be reformatted, you need to use the System Menu on its storage node.

> Note
> This process requires the chassis to be taken offline.

1. Using a remote terminal or a keyboard and monitor, access the System Menu on the Swarm storage node.
2. Take the chassis offline.

   a. Select 4. System Control.
   b. Select 3. Stop Storage Processes.
   c. Confirm by choosing Yes. At that point, you should see Storage Processes: STOPPED
3. Mark the drive for reformatting on next reboot.
   a. Select 2. Disk Volumes
   b. Arrow down to the drive in question and press "`f`"
   c. When prompted, type `FORMAT` to continue. (This only queues the formatting, so do not expect a delay.)
4. Restart the chassis.
   a. Select 4. System Control.
   b. Select 1. Reboot System.

Reformatting all Drives in a Cluster

To reformat all drives in a cluster:

1. Shut down all nodes in the cluster.

   > Note
   > To reformat an entire cluster, you will reformat one node at a time. Otherwise, the remaining nodes in the cluster will begin replication and move objects to your reformatted node.

2. Remove the USB flash drive from any node in the cluster.
3. (versions prior to 6.0) Edit the configuration file and remove the cluster.settingsUuid value.
4. Insert the USB flash drive in the node and reboot it.
5. In the System Information menu, select Disk Volumes.
6. In the Disk Volumes menu, select a formatting option.
7. Wait for the selected drives to be formatted.
8. Shut down the node.
9. Repeat step 2 through step 8 on the remaining cluster nodes, one node at a time.
10. After all nodes are reformatted, start the nodes in any order.

Configuring Swarm for Gateway

- Network Placement
- Domain Management
- Elasticsearch Servers
- Configuration Requirements

This section provides information specific to running Swarm Storage with Gateway. First, you need to install and configure Swarm, the storage cluster (storage nodes are appliances that run on dedicated hardware).

Network Placement

When deployed with Gateway, the storage nodes should be placed on a network subnet that is not directly accessible to the client applications. This way, all user communications with the storage cluster must go through the Gateway.

> Caution

> If users are allowed to communicate directly with the storage cluster nodes, they may bypass access security, the business rules for content metadata, and audit logging that is performed by the Gateway and may render content in the cluster unusable to the Gateway. Only allow direct access to the storage cluster nodes under highly controlled circumstances, such as administrator-only operations or trusted applications.

Domain Management

The Swarm cluster provides for logical separation of content among multiple tenants through the use of storage domain names. Gateway has the following requirements above and beyond those for a baseline storage deployment and client usage.

- An administrative domain must be created in the storage cluster.
- At least one storage domain must be created in the storage cluster.
- Storage domains must adhere to IANA naming standards (that is, be valid DNS names).
- Client applications must specify a storage domain in every request.

The storage domain name for an operation is specified by the client application according to the following precedence from highest to lowest:

- SCSP `domain=X` query argument
- HTTP `X-Forwarded-Host` header
- HTTP `Host` header

In order to make use of the Host header to identify the storage domain with most HTTP/1.1 libraries, storage domains in Swarm must resolve to least one IP address ("A" record) for client applications. Additionally, the resolved IP address should be for a Gateway or, if applicable, some other front-end network appliance such as a load balancer. If there are multiple Gateway servers, using a DNS round-robin with their IP addresses is a valid configuration to use.

This is an example of a BIND 9 zone file that implements a wildcard of all storage domains within the `cloud.example.com` parent DNS domain and points them to the IP address `10.100.100.100`.

```
$TTL 600 @ IN SOA cloud.example.com. dnsadmin.example.com. (
    2016070201 ; Serial number
    4H     ; Refresh every 4 hours
    1H     ; Retry every hour
    2W     ; Expire after 2 weeks
    300 )  ; nxdomain negative cache time of 5 minutes
IN NS ns1.example.com.
* IN A 10.100.100.100
```

In the example zone file, `10.100.100.100` is the IP address used by client applications to communicate with the Gateway or a front-end load balancer. The names `hydrogen2.cloud.example.com` and `oxygen.cloud.example.com` would both resolve to the same IP address.

Elasticsearch Servers

When using the S3 storage protocol, the metadata search service must be accessible to the Gateway servers.

When deployed with Gateway, like the storage nodes, the typical placement will be on a network subnet that is not directly accessible to the client applications. At this time, there are no end-user supported API calls directly to the metadata search service.

> Best practice
> When defining the search feed for use with Gateway, set the batch timeout value to a low value (such as 1 to 5 seconds) in order to provide good interactive search-after-create response to the client applications that use the Gateway.

Configuration Requirements

When using Swarm Storage with Gateway, you must use these Swarm configuration settings and adhere to the following operational changes. These configuration changes refer to the configuration file(s) for Swarm. This is either the `node.cfg` file or, when using a Platform Server to manage the nodes, the global configuration file for the cluster:

```
/var/opt/caringo/netboot/content/cluster.cfg
```

> Caution
> Failure to use these settings and operational changes can prevent Gateway from working properly with the storage cluster.

Requirement	Description

Enable `ecEncoding`	You must enable and configure erasure coding; if not, the Gateway cannot start (`service cloudgateway init` will fail).
	`[policy]`
	`ecEncoding = {k:p}`
	`ecMinStreamSize = 1MB`
	See Implementing EC Encoding Policy.

Enable `enforceTenancy`	Enforce the use of tenancy for unnamed objects by setting:
	`[cluster]`
	`enforceTenancy = True`

<table>
<tr>
<td>Enable noauth</td>
<td>

Be sure to disable the legacy Swarm authentication/authorization by setting:

```
[security]
noauth = True
```

</td>
</tr>
<tr>
<td>Storage Domain Management</td>
<td>

Only create and manage storage domains through the Content Portal or programmatically through the Gateway's management API.

> Important
> When using Content Gateway, do not use the legacy Admin Console (port 90) to manage storage domains.

Troubleshooting: A domain created by the legacy Admin Console (port 90) contains the legacy Castor-Authorization header, which causes the Content UI to report "Page Not Found: The original bucket to which this collection refers cannot be found or has been replaced..." The fix is to remove the Castor-Authorization header (which removes Swarm access control and assumes your Swarm nodes are accessible only to trusted clients) and ensure that "caringoadmin" is a Gateway LDAP or PAM user (idsys.json). To remove the header, issue a COPY request directly to a Swarm node (the semicolon syntax to remove header values requires a curl version later than 7.29).

```
curl -v -u admin --post301 --location-trusted -XCOPY
    -H 'X-Owner-Meta: caringoadmin'
    -H "content-type:application/castorcontext"
    -H 'Castor-Authorization;'
    'http://SCSP_HOST/?domain=myolddomain&preserve&admin'
```

</td>
</tr>
</table>

Drive Identification Plugin

- Drive Light Customization
- Drive Light API

Swarm supports a drive identification function that flashes an LED light next to a volume that you select in the Swarm UI (or legacy Admin Console). The plugin also turns on the drive light whenever a disk fails to mount, whether at reboot or hot-plug event.

The Identify function is implemented as a Linux shell script daemon. When the Identify function is enabled, Swarm tries one of several tools (sg_ses, MegaCli64, sas2ircu, sas3ircu) to turn on the identification light. If none of those tools succeed, Swarm will continuously read blocks from the selected drive to force the I/O activity drive light to flash.

- The script is provided in the Swarm distribution: `caringo/samples/genericDrivePlugin.sh`

- The script is overridden if there is a custom value in the disk.volumeIdentifyFiles setting.

> Best practice
> Use the generic plugin and remove any disk.volumeIdentifyFiles value from your configuration files.

Drive Light Customization

Integrators or hardware vendors with knowledge of manufacturer-specific methods for flashing drive light may substitute an alternative method using the plugin drive identification API.
To customize drive light script:

1. Open the Swarm software distribution and navigate to the `caringo/samples` directory.
2. In the directory, copy the genericDrivePlugin.sh script to a new directory.
3. Rename the script with a unique name that does not contain any periods or special characters.
4. Open the script and modify the identifyOn() and identifyOff() functions as required to implement an alternate drive light identification mechanism.
5. Archive (tar) and compress (gzip) the script and any other required custom files.
6. Deploy the script.
7. Place the tar file on an HTTP server or Swarm USB drive.
8. Update the disk.volumeIdentifyFiles setting to point to the location of your tar file.

Drive Light API

These are the key functions in the generic script:

identifyOn()	Turns on the drive identification light.
identifyOff()	Turns off the drive identification light.
_monitorAndInitiate()	Polls for a drive status file and initiates identification by spawning _identifyOn().
_identifyOn()	Spawned as a process when identification is turned on by Swarm for a volume (for example, one process per volume). It calls the _identifyOn() function and then waits for identify to get turned back off by Swarm so that it can call the _identifyOff() function.

Swarm Passwords

The Swarm `admin` password is required to make changes in the Swarm Storage UI; read-only credentials (`operators`) are needed to view it. See Defining Administrators and Users.

> Important
> When you deploy your storage cluster, be sure to change the default passwords.

As of Swarm 10.0, there are two pairs of passwords to manage, which are in the cluster-wide `[security]` and `[snmp]` sections of the Swarm Storage configuration file (`cluster.cfg` (Platform/CSN) or else `node.cfg):`

Setting name	Default	Descriptions and Examples
security.administrators	`{'admin': 'ourpwdofchoicehere'}`	One or more username:password pairs. Sets credentials for who can administer the cluster via the Swarm UI. Upgrading from 9.x If the value includes the snmp username, remove it from here and update `snmp.rwCommunity` with its password. Example: `{'admin': 'adminpassword','admin2':'adminpassword2'}`
security.operators	`{}`	One or more username:password pairs. Sets credentials for who can view the Swarm UI. Upgrading from 9.x If the value includes an snmp username, it is ignored; remove it from here and update `snmp.roCommunity` with its password. Example: `{'operator': 'operatorpassword','operator2': 'operatorpassword2'}`
snmp.rwCommunity	`ourpwdofchoicehere`	String. Password for the SNMP read-write community. Important You must know the SNMP read-write password in order to dynamically change the Swarm '`admin`' password. To change the SNMP read-write password, you must edit the config file.
snmp.roCommunity	`public`	String. Password for the SNMP read-only community.

Caution

- The name **`admin`** is reserved, so do not delete it, which could cause errors and affect performance. If you decide not to use **`admin`**, define a complex password to protect it.
- Swarm prevents cluster booting if the SNMP security administrator (read/write user) is not set properly in the configuration file.
- All administrative users and passwords must agree on all nodes or certain cluster actions will fail.
- Password updates are not complete until they are persisted in the cluster settings file across all nodes, and rapid, successive updates cannot be accepted on a given node until the first update completes

processing.

Encrypting Passwords

Instead of a clear text password, you can represent the password as a hexadecimal-encoded MD5 hash of the following string:

```
username:user-list-name:password
```

where username and password consist only of ASCII characters and `user-list-name` can be either "CAStor administrator" or "CAStor operator".

To create the MD5 hash, use a programming language or a utility such as md5sum or Apache htdigest. For example, to update your node or cluster configuration file with a password hash you create using htdigest:

1. Create a file that contains a hash of the user name, password, and user list name:

   ```
   htdigest -c password-file.txt "CAStor administrator" Jo.Jones
   ```

2. When prompted by htdigest, enter and confirm the user's password.
3. Open the new file (`password-file.txt`) in a text editor. The hash is the last entry in the string:

   ```
   Jo.Jones:CAStor administrator:08b0468c1d957b7bac24463dd2191a2d
   ```

Updating Passwords

How you update the passwords depends on which ones need updating and whether Swarm has ever been started.

> Note
> If you do not use a Platform Server, make these changes to the `node.cfg`.

Situation	Process	Examples and notes

<table>
<tr>
<td>Swarm has never booted</td>
<td>

1. Create and hash an `admin` password.
2. Update passwords in the config file.
3. Important: If booting from a USB flash drive, be sure to unmount/stop the USB drive or else the changes will not be saved.
4. Boot the Swarm cluster.
5. After the cluster is running, you can remove the password from the config file.

</td>
<td>

Hash of password

```
$ echo -n 'admin:CAStor
administrator:NEWPASSWORD' | md5sum | cut -d
' ' -f1
7fe563b8532b3a460def0895895eebf5
```

The first time that you boot the cluster, the Swarm admin password must be in the config file:

```
[security]
administrators =
{'admin':'7fe563b8532b3a460def0895895eebf5'}
```

When the cluster is running, Swarm stores the admin password in the persisted Settings object, at which point it is safe to remove the password from your configuration file for security purposes:

```
[security]
administrators = {}
```

</td>
</tr>
<tr>
<td>Updating SNMP passwords</td>
<td>

1. Update passwords in the config file.
2. Reboot the Swarm cluster.

</td>
<td>

Important

You must know the SNMP read-write password in order to dynamically change the Swarm '`admin`' password. If you need to change the SNMP read-write password, you must edit the config file.

After rebooting with the new SNMP password in the file, you can proceed to change the Swarm '`admin`' password.

</td>
</tr>
</table>

Updating Swarm admin password	1. Create and hash an admin password.
	2. Update password via SNMP, which Swarm will save in the persisted Settings object.
		Changing admin password
		`snmpset -v2c -c SNMP- password -m`
		`+CARINGO-CASTOR-MIB Swarm- node- IP`
		` addModifyAdministrator s "admin:new-`
		`password"`
		
		`snmpset -v2c -c ourpwdofchoicehere -m`
		`+CARINGO-CASTOR-MIB 172.20.3.85`
		` addModifyAdministrator s`
		`"admin:7fe563b8532b3a460def0895895eebf5"`

Frequently asked questions:

- How do I change the active SNMP read-write password? The SNMP passwords cannot be changed dynamically. Changing one or both requires a config file update and a cluster reboot.
- What is the SNMP read-only password? The read-only password 'public', which is the 'community string'
- Is the read-only SNMP password in the persisted Settings object? No
- Can my SNMP read-write passwords in the persisted Settings object and cluster.cfg be different? Yes, but only the config file SNMP read-write password is used.
- How do I change my admin password? Update the password via SNMP and then update it in the config file, unless you've removed it from there.
- How do I change my SNMP read-only password to the cluster? Change the snmp.roCommunity setting in the config file and reboot the cluster.

Elasticsearch Implementation

Swarm integrates Elasticsearch and extends the Swarm API with commands for querying Swarm objects in terms of their metadata. Through this feature, Swarm indexes object metadata in near real time and lets you perform ad hoc searches (via query commands) on the attributes and metadata of your stored objects.

Swarm uses Elasticsearch servers for its metadata searching operations. You can deploy these servers for high-availability and horizontal scaling. Although high availability of the search cluster is not needed for high availability of the storage cluster, you may need it to service third-party analytics applications.

> **Important**
> For production-level responsiveness and redundancy, deploy at least three search servers. Be sure to follow the Hardware Requirements for Elasticsearch.

You can return the results as JSON or XML, which you can import into your third-party analytics applications.

> See also these sections:
>
> - Swarm Storage Release Notes
> - Elasticsearch for Swarm (configuration and administration)

> - Swarm Historical Metrics
> - Storage SCSP Development

Search components

The search infrastructure includes these components:

- Swarm Storage cluster, which is connected to the Elasticsearch servers through a Search Feed.
- Search feed(s), which transmit the metadata from the storage cluster. Feeds iterate over data on storage nodes and use intermittent channel connections to distribute data to one or more configured destinations, including metadata search servers. See Viewing and Editing Feeds.

  > Tip
  >
  > Because Swarm uniquely names each search feed index, you can configure additional feeds that use the same Elasticsearch cluster; however, be sure to plan for doubling or tripling the space demands on that server.

- Elasticsearch servers, which index the metadata and service search requests. This metadata can be reconstructed from the storage cluster, if needed.
- Metrics curator service, which can be installed on one of the Elasticsearch servers, or another system running RHEL/CentOS 7.
- Client applications, which access the Swarm cluster through SCSP commands.

> Best practice
>
> Devote the search cluster to Swarm-only usage, and do not store non-Swarm data in your search installation.

- Hardware Requirements for Elasticsearch
- Preparing the Search Cluster
- Installing Elasticsearch
- Migrating from Elasticsearch 2.3.3
- Configuring Elasticsearch
- Installing Swarm Metrics

Hardware Requirements for Elasticsearch

- Scaling Elasticsearch
- Hardware Best Practices
- RAM for Elasticsearch
- Disk Usage for Search
- Optimizing Disk I/O for ES
- Optimizing Disaster Recovery for ES

Your Elasticsearch cluster supports both Swarm searches and Historical Metrics. The Swarm Feeds mechanism popula tes the metadata search servers that run the Elasticsearch (ES) software.

> See Elasticsearch Implementation and Installing Swarm Metrics.

This software requires one or more servers running RHEL/CentOS 7 Linux. Although Elasticsearch runs on other Linux platforms, we currently provide support and testing for these versions. Only the Elasticsearch version provided with the Swarm distribution is supported.

> See the Elasticsearch project website for more information about Elasticsearch.

Scaling Elasticsearch

The hardware platform for the Elasticsearch servers can scale from one virtual machine to multiple physical machines.

> Best practice
> Elasticsearch recommends not exceeding 200 million documents (unique objects) per ES node. Staying within the limit for number of documents per node maximizes performance.

Scale	ES nodes	Max documents	Swarm nodes	Metrics retention	Notes
Minimum	1	200M	3		ES 5.6: Run script to set replicas to zero (see below). Do not use without a robust ES backup strategy
Minimal	2	400M	≤ 50	≤ 90 days	Not recommended; lower performance than Basic
Basic	3	600M	≤ 100	≤ 120 days	
Large	5	1000M	≤ 200	≤ 240 days	
Very large	8	1600M	> 200	≤ 365 days	

> For details about the Metrics templates, see Installing Swarm Metrics.

Replicas in ES 5.6 — Elasticsearch 5.6 no longer allows you to configure the number of replica shards for new indices in the `elasticsearch.yml` file. Instead, it provides a default replica value of 1, which works for most installations. However, if you are configuring an Elasticsearch cluster with only one Elasticsearch data node, then all your indices will have yellow status, because the primary and replica shards should be hosted on separate nodes. In that case, configure your indices to have zero replicas, using the script provided:

```
/usr/share/caringo-elasticsearch-search/bin/configure_replicas.py -r 0
-e <ES-SERVER>
```

To view the complete options for changing the number of replicas on existing indices, use the help command:

```
/usr/share/caringo-elasticsearch-search/bin/configure_replicas.py --help
```

Replicas in ES 2.3.3 — By default, Elasticsearch allocates 5 primary shards and one set of replicas per index. With three or more ES servers, this provides high availability. With five or more servers, you can better spread the transactions across multiple servers, providing additional processing power.

- Swarm Search indices use the default 5 shard/1 replica scheme. If you need higher availability and query throughput, configure additional replicas by editing the "index.number_of_replicas" property in the elasticsearch.yml file.
- Swarm Metrics indices are configured with 1 shard and one additional replica per index. Additional replicas can be configured by editing the "index.number_of_replicas" property in the elasticsearch.yml file. If more shards are desired, contact Support for help with that configuration.

Hardware Best Practices

Following are overall best practices, with hardware recommendations from Elasticsearch:

- Provision the machines with CPUs that have at least 4 cores and 64 GB memory. Between faster processors or more cores, choose more cores.
- Choose SSD drives, to boost performance.
- If using hard disk drives (which do not handle concurrent I/O as well as SSDs), do the following:
  - Select high-performance server drives.
  - Use RAID 0 with a writeback cache.
  - Set `index.merge.scheduler.max_thread_count` to 1, to prevent too many merges from running at once.

```
curl -X PUT <ES_SERVER>:9200/<SWARM_SEARCH_INDEX>/_settings \
 -d '{ “index”: {“merge.scheduler.max_thread_count”: 1}}'
```

- As with your storage cluster, choose similar, moderate configurations, for balanced resource usage.

RAM for Elasticsearch

RAM is key for Elasticsearch performance. Use these guidelines as a basis for capacity planning:

- 64 GB RAM per machine is optimal (recommended by Elasticsearch).
- Dedicate half of total RAM to the Java Virtual Machine (JVM) that runs Elasticsearch, but do not exceed 31 GB, for best performance.
- For ES 5.6.x, disable swapping of the Elasticsearch image. (For ES 2.3.3, allow in-memory caching of all shards on the server.)

You can achieve optimal performance by adding adequate RAM in your ES servers to store all database shards in memory. Take steps to disable or mitigate swapping. If memory page swapping begins to occur on an ES server, it will impact Elasticsearch performance.

> Important
>
> When monitoring your ES servers, watch for sustained increases in page swapping and disk I/O, which might mean that you need to add RAM to an ES server or deploy additional servers to offset the load.

Disk Usage for Search

The storage on the Elasticsearch servers is used to persist the shards of the Swarm Search and Swarm Metrics indices. Follow these guidelines for capacity planning for the Swarm Search indices.

- Assumption: 2 KB average per metric sample × 7 metrics indices × 4 samples/hour × 120 days data retention ≈ 161 MB per Swarm node being indexed
- Baseline metadata to support listing only: 150 GB per 200 million objects
- Full metadata to support ad-hoc searching: 300 GB per 200 million objects
- Custom metadata: if you index a large amount of custom metadata, allocate additional storage in proportion

These are objects, not replicas: how many Swarm replicas a Swarm object has is irrelevant to the ES servers. No matter how many replicas of an object exist in the cluster, there will be only one metadata entry for the object.

> Tip
>
> Do not confuse this with the RAM-based Overlay Index that each storage node maintains, which depends on the total number of replicas in the cluster.

Optimizing Disk I/O for ES

Elasticsearch makes heavy use of drives, so higher throughput means more stable nodes. Follow these Elasticsearch guidelines for optimizing disk I/O:

- Use SSDs. SSDs will boost performance. With SSDs, ensure that your OS I/O scheduler is configured correctly.
- Use RAID 0. Striped RAID will increase disk I/O, at the expense of potential failure if a drive dies. Do not use mirrored or parity RAIDS, because replicas provide that functionality. Alternatively, use multiple drives and allow Elasticsearch to stripe data across them via multiple path.data directories.
- Do not use remote-mounted storage, such as NFS or SMB/CIFS; the latency will hurt performance.
- Avoid virtualized storage, such as EBS (Amazon Elastic Block Store). Even SSD-backed EBS is often slower than local instance storage.

Optimizing Disaster Recovery for ES

Elasticsearch clustering has been specifically designed to mitigate the impact of hardware and power failures, so that you do not experience long delays from refreshing the search data. How much you should invest to optimize your hardware depends on how important metadata search and querying is to your organization and how long these features can be offline while Elasticsearch rebuilds its data.

These are principles for making your configuration more disaster-proof:

- Do not use and rely on just a single Elasticsearch server. This makes your search and metrics capabilities vulnerable to disruption, and it risks too little capacity to support all your search and metrics needs.
- For power failure protection, deploy enough Elasticsearch servers to survive multiple server failures and distribute them across different power sources.

- If your cluster is divided into subclusters to match your power groups, then set up Elasticsearch with multiple nodes that are spread equally among the subclusters. This strategy improves survivability of a power group failure.

Preparing the Search Cluster

To prepare the search servers for Elasticsearch, perform the following steps.

1. Verify your servers against the Hardware Requirements for Elasticsearch.
2. Appropriately cable the servers to your network infrastructure so that, once configured, they will be reachable from the Swarm nodes.
3. Install RHEL/CentOS 7 Linux and apply any required updates.

   > Best practice
   >
   > Use RHEL/CentOS Minimal Server (with Compatibility Libraries), which is the standard for Caringo development and testing. RHEL/CentOS Desktop consumes extra resources that Elasticsearch could use, alters the OS configuration to emphasize user interface vs. server performance, and requires additional updating and security maintenance.

4. Configure the servers with static IP addresses.
5. Configure DNS, if desired.
6. Adjust the server firewall rules. (See RHEL/CentOS documentation for the iptables syntax.) You may disable iptables to allow required network communications. However, if you are running iptables, you must adjust the rules to permit the following ports:
   - `9200/tcp`
   - `9300/tcp`
   - `54328/udp`
7. Install Java using this command:

   ```
   yum install java-1.8.0-openjdk
   ```

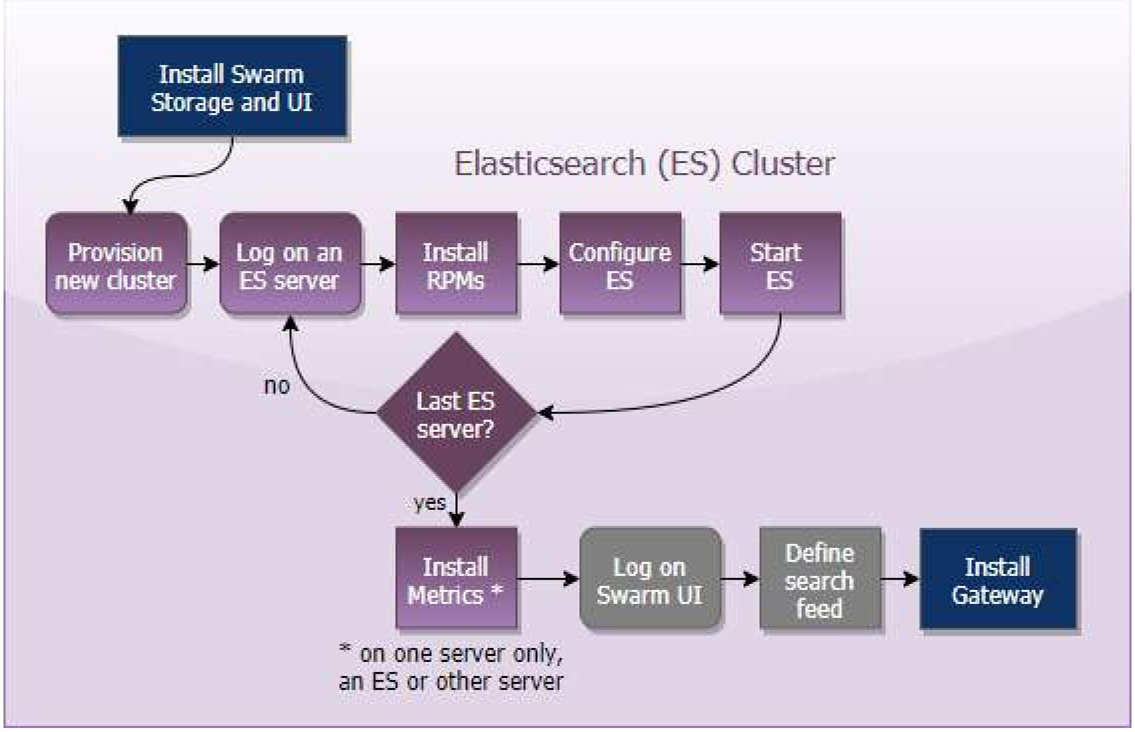
   > Required
   >
   > Swarm requires Java 8 (update 20 or later). The JDK must be Oracle's Java or the OpenJDK. Use the same JVM and version on all of your search nodes and clients.

8. After installing, be sure to select the correct version of Java. An incorrect Java version could cause a failure.

```
alternatives --config java
Selection    Command
-----------------------------------------------
*  1           /opt/jdk1.8.0/bin/java
 + 2           /opt/jdk1.8.0_25/bin/java
   3           /opt/jdk1.8.0_40/bin/java
```

Installing Elasticsearch

This is an overview of the Elasticsearch (ES) installation process:



1. Prepare for installation.
    a. From the Swarm bundle download, get the latest Elasticsearch RPM and Swarm Search RPM, which installs plugins and support utilities.

    ```
    elasticsearch-<VERSION>.rpm
    caringo-elasticsearch-search-<VERSION>.noarch.rpm
    ```

    b. If you have not previously added the Caringo RPM public key that is included with the distribution bundle to your system, run the following command to do so:

```
rpm --import RPM-GPG-KEY
```

   c. If you are already running Elasticsearch with Swarm, pause your feed and stop the service before proceeding:

```
systemctl stop elasticsearch
```

2. On each ES server, install and configure the Elasticsearch components.
   a. Install the RPMs.

```
yum install elasticsearch-<VERSION>.rpm
yum install caringo-elasticsearch-search-<VERSION>.noarch.rpm
```

   b. Complete configuration of Elasticsearch and its environment. See Configuring Elasticsearch.
   The configuration script will start the Elasticsearch service and enable it to start automatically.
   c. Verify that the mlockall setting is true. If it is not, contact your Support representative.

```
curl -XGET "<ES_HOST>:9200/_nodes/process?pretty"
```

   d. Proceed to the next server.
3. At this point, all ES servers should be installed and started. Use one of these methods to verify that Elasticsearch is running (the status is yellow or green):

```
curl -XGET <ES_HOST>:9200/_cluster/health
```

```
systemctl status elasticsearch
```

> Tip
> When troubleshooting Elasticsearch issues, run the status command (`systemctl status elasticsearch)` and then look at the log entries:
> `/var/log/elasticsearch/<clustername>.log`

4. Install Swarm Metrics on just one server in the Elasticsearch cluster (or another machine running RHEL/CentOS 7).
   See Installing Swarm Metrics.
5. In the Swarm Storage UI (or legacy Admin Console), create a new search feed. (See Viewing and Editing Feeds.) Indexing is done when the feed shows 0 "pending evaluation".

> Tip
> To set up the ability to restore your search data on demand, see Snapshot and Restore Search Data.

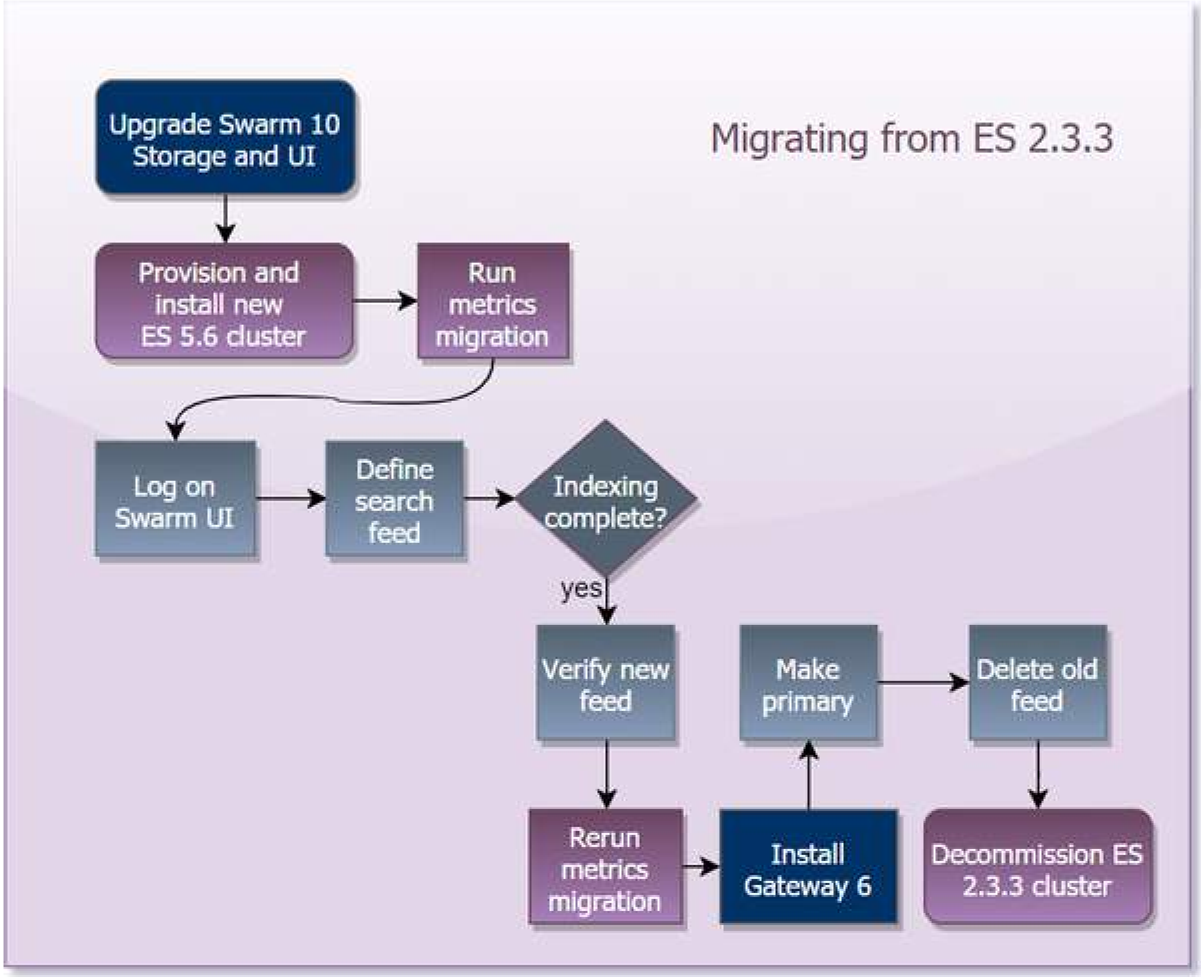
Migrating from Elasticsearch 2.3.3

- Pre-Upgrade Checklist
- Install Elasticsearch 5.6
- Migrate Metrics Data
- Switch to ES 5.6

Elasticsearch 5.6 includes advances in speed, security, scalability, and hardware efficiency and supports newer tool releases. All upgrades between major release numbers of Elasticsearch are considered migrations that require both a cluster restart and reindexing of prior data.

Given the complexities of converting legacy ES (Elasticsearch) data, the easiest path is to start fresh: provision a new E S cluster (machines or VMs that meet the requirements), install the ES and Search RPMs on those servers, and create a new search feed to this cluster. Swarm will continue to support your old primary feed with the ES 2.3.3 cluster while it builds the index data for the new feed, which means that searching remains available for your users. Once the new feed is built, you make it the primary and the migration is complete.

This is an overview of the migration process:



Exceptions

This upgrade strategy does not fit certain situations:

- You cannot provision a new search cluster.
- You use parallel upload in production and need support for uploads to complete during the migration phase (uploads of parts are unaffected).

For such cases, contact Support for assistance.

- Pre-Upgrade Checklist
- Install Elasticsearch 5.6
- Migrate Metrics Data
- Switch to ES 5.6

Pre-Upgrade Checklist

Swarm Requirements	1. Complete the upgrade of Swarm to 10.0. This version maintains compatibility support for Elasticsearch 2.3.3 and your existing indices/feeds, but new features in Swarm 10.0 are unavailable until you complete the Elasticsearch migration. 2. Case-sensitivity - If you enable case-insensitive searching in SCSP (`search.caseInsensitive = 1`) , Content Gateway still lets S3 perform the case-sensitive operations it needs.
New ES Cluster	- Provision a new set of ES servers (machines or VMs) on which to install Elasticsearch 5.6. - Ensure that every Elasticsearch node meets the hardware, network, and software requirements, including RHEL/CentOS 7 and Java 8. See Hardware Requirements for Elasticsearch. See Preparing the Search Cluster.

Install Elasticsearch 5.6

> **Important**
> Never run different versions of Elasticsearch in the same cluster. Ensure that your new Elasticsearch configuration has a different name for the 5.6 cluster; otherwise, the new ES servers will join the old ES cluster.

1. Prepare for installation.
   a. From the Swarm bundle download, get the latest Elasticsearch RPM and Swarm Search RPM, which installs plugins and support utilities.

   ```
   elasticsearch-<VERSION>.rpm
   caringo-elasticsearch-search-<VERSION>.noarch.rpm
   ```

   b. If you have not previously added the Caringo RPM public key that is included with the distribution bundle to your system, run the following command to do so:

   ```
   rpm --import RPM-GPG-KEY
   ```

   c. If you are already running Elasticsearch with Swarm, pause your feed and stop the service before proceeding:

   ```
   systemctl stop elasticsearch
   ```

2. On each ES server, install and configure the Elasticsearch components.
   a. Install the RPMs.

```
yum install elasticsearch-<VERSION>.rpm
yum install caringo-elasticsearch-search-<VERSION>.noarch.rpm
```

b. Complete configuration of Elasticsearch and its environment. See Configuring Elasticsearch.
The configuration script will start the Elasticsearch service and enable it to start automatically.

c. Verify that the mlockall setting is true. If it is not, contact your Support representative.

```
curl -XGET "<ES_HOST>:9200/_nodes/process?pretty"
```

d. Proceed to the next server.

3. At this point, all ES servers should be installed and started. Use one of these methods to verify that Elasticsearch is running (the status is yellow or green):

```
curl -XGET <ES_HOST>:9200/_cluster/health
```

```
systemctl status elasticsearch
```

> Tip
> When troubleshooting Elasticsearch issues, run the status command (`systemctl status elasticsearch)` and then look at the log entries:
> `/var/log/elasticsearch/<clustername>.log`

4. Install Swarm Metrics on just one server in the Elasticsearch cluster (or another machine running RHEL/CentOS 7).
See Installing Swarm Metrics.
5. In the Swarm Storage UI (or legacy Admin Console), create a new search feed. (See Viewing and Editing Feeds.)
Indexing is done when the feed shows 0 "pending evaluation".

> Note
> Swarm Storage lets you create more than one Search feed so that you can transition from using one Elasticsearch cluster to another. During the transition, continue using the primary feed for queries; the second feed is incomplete until it fully clears its backlog. When the second feed is caught up, transition to it (marking it as primary) as soon as reasonable for your operations. When you verify that it is working as the new primary feed target, delete the original feed. Having two feeds is for temporary use only because every feed incurs cluster activity, even when paused.

Migrate Metrics Data

Any existing metrics indices need to be migrated to the new schema format. The Swarm Metrics RPM ships with a script that migrates existing metrics data from the Elasticsearch 2.3.3 cluster to your new Elasticsearch 5.6.x cluster.

1. Install Swarm Metrics in your new Elasticsearch cluster. See Installing Swarm Metrics.
2. Before you run the migration script, make sure to "whitelist" the Elasticsearch server (running ES 5.6), so that it trusts the source Elasticsearch server.
   - On the destination ES node, edit the config file: `/etc/elasticsearch/elasticsearch.yml`
   - Add the whitelist line:

     ```
     reindex.remote.whitelist: <SOURCE_ES_NODE>:<SOURCE_ES_PORT>

     reindex.remote.whitelist: indexer1.tx.caringo.com:9200
     ```

3. The migration script is written in Python and uses the Requests package, so ensure that these are installed before running the script.
4. Run the script, specifying your source and destination clusters:

   ```
   /usr/share/caringo-elasticsearch-metrics/bin/reindex_metrics -s
   <ES_2_SERVER> -d <ES_5_SERVER>
   ```

   Legacy csmeter — If you have legacy csmeter data and you do not want to migrate that data, add a final `-c` flag when you invoke the script.
5. Allow an hour or more for the script to complete if you have a large amount of metrics to convert (many nodes and several months of data).
6. If connection or other problems occur and the screen reports errors, run the script again, and repeat until it completes successfully.
7. Reconfigure the Swarm setting for metrics to begin sending metrics data to new your ES 5.6 cluster. Add your new metrics server to `metrics.targets`.
   See step 5 of Installing Swarm Metrics.
8. If metrics had already been running when you first ran the migration script, run it one more time with the "-f" flag, which forces it to merge old metrics data with the new metrics data:

   ```
   /usr/share/caringo-elasticsearch-metrics/bin/reindex_metrics -s
   <ES_2_SERVER> -d <ES_5_SERVER> -f
   ```

Switch to ES 5.6

1. Wait until the new search feed has completed indexing the cluster, when the feed shows 0 "pending evaluation".

> Tip
> Set calendar reminders to check on the progress. How many days this takes depends on the cluster's size and load, and you could forget to finish this upgrade.

2. When the new feed is ready and verified, select Make Primary.
3. Operate with both search feeds for several days. If there is a problem, you can restore the old feed to be primary during troubleshooting.
4. To ensure that all the metrics data for the current day is available, run the migration script one last time, using the "-f" flag, which forces reindexing of the current day's data.

```
usr/share/caringo-elasticsearch-metrics/bin/reindex_metrics -s
<ES_2_SERVER> -d <ES_5_SERVER> -f
```

5. After this confirmation period, delete the old feed.
6. At this point, upgrade Gateway. See Gateway Installation.
7. Decommission your ES 2.3.3 cluster to reclaim those resources.

Configuring Elasticsearch

Elasticsearch requires configuration and settings file changes to be made consistently across your Elasticsearch cluster.

1. On each of your Elasticsearch nodes, run the provided configuration script (`/usr/share/caringo-elasticsearch-search/bin/configure_elasticsearch_with_swarm_search.py`), which automates the configuration changes described below.

If you do not need any of the customization below, skip this section and resume your installation to turn on the service: Installing Elasticsearch

1. If you need to customize any settings, such as to change Elasticsearch's `path.data` (data directory), proceed as follows below.

> Important
> Be sure to update log files to match your customizations.

2. Choose one of these methods:
   - Run the script the same way on each ES server, answering all of the config questions identically (hosts, cluster name, etc).
   - Run the script on one node, which will generate an elasticsearch.yml file for each Elasticsearch server. Copy those yaml files to the remaining Elasticsearch servers and then run using the "-c" option, which uses the elasticsearch.yml file and its config settings but customizes all other configuration-related files:

```
configure_elasticsearch_with_swarm_search.py -c
/etc/elasticsearch/elasticsearch.yml
```

Customizing Elasticsearch

- Elasticsearch Config File
- Systemd (RHEL/CentOS 7)
- Environment Settings
- JVM Options
- Log Setup for 5.6
- Log Setup for 2.3.3
- Log Rotation

The paths given are relative to the Elasticsearch installation directory, which is assumed to be your working directory.

> Caution
>
> - Errors in adding and completing these settings can prevent the Elasticsearch service from working properly.
> - If you customize the Elasticsearch's `path.data` location from the default, you must adjust all references to it below to reflect the new location.

Elasticsearch Config File

Edit the Elasticsearch config file: `/etc/elasticsearch/elasticsearch.yml`

`cluster.name: <ES·cluster·name>`	Give your Elasticsearch cluster a unique name, which is unrelated to your Swarm cluster name. Do not use periods in the name. Important To prevent merging, it must differ from the `cluster.name` of your ES 1.7.1 or ES 2.3.3 cluster, if you have one.
`node.name: <ES·node·name>`	Optional. Elasticsearch will supply a node name if you do not set one. Do not use periods in the name.

`network.host: <ES·host>`	Assign a specific hostname or IP address, which requires clients to access the ES server using that address. If you use a hostname, update `/etc/hosts`. Defaults to the special value, `_site_`. Metrics requirement If you configure the Elasticsearch host to a specific hostname or IP address, then the Elasticsearch host for Metrics in /etc/caringo-elasticsearch/metrics/metrics.cfg must match. However, if you configure `network.host` in elasticsearch.yml to be "`_site_`", then the host in metrics.cfg can be a valid IP address or hostname for that Elasticsearch server.
`discovery.zen.ping.unicast.hosts: ["es2", "es3"]`	Set to the list of node names/IPs in your cluster, making sure to include all of your ES servers. By default, multicast is disabled.
`discovery.zen.minimum_master_nodes: 3`	Set to (number of master-eligible nodes / 2, rounded down) + 1 Prevents split-brain scenarios by setting the minimum number of ES nodes that must be online before deciding on electing a new master.
`gateway.expected_nodes: 4`	Add and set to the number of nodes in your ES cluster. Recovery of local shards will start as soon as this number of nodes have joined the cluster. It falls back to the `recover_after_nodes` value after 5 minutes. This example is for a 4-node cluster.
`gateway.recover_after_nodes: 2`	Set to the minimum number of ES nodes that must be started before going into operation status. This example is for a 4-node cluster.
`index.max_result_window: 50000`	(Elasticsearch 2.3.3. only) Add to support queries with very large result sets (it limits start/from and size in queries). Elasticsearch accepts values up to 2 billion, but more than 50,000 consumes excessive resources on the ES server.
`index.translog.sync_interval: 5s`	(Elasticsearch 2.3.3. only) For best performance, set how often the translog is fsynced to disk and committed, regardless of write operations.
`index.translog.durability: async`	(Elasticsearch 2.3.3. only) For best performance, change to `async` so that ES will fsync and commit the translog in the background every `sync_interval`. In the event of hardware failure, all acknowledged writes since the last automatic commit will be discarded.

`bootstrap.mlockall: true`	(Elasticsearch 2.3.3 only, in place of `bootstrap.memory_lock`) Set to lock the memory on startup to ensure that Elasticsearch never swaps (swapping makes it perform poorly). Ensure that enough system memory resources are available for all processes running on the server.
	To allow the `elasticsearch` user to disable swapping and to increase the number of open file descriptors, the RPM installer makes these edits to`/etc/security/limits.d/10-caringo-elasticsearch.conf`:
	`# Custom for Caringo Swarm`
	`elasticsearch soft nofile 65535`
	`elasticsearch hard nofile 65535`
	`# allow user 'elasticsearch' mlockall`
	`elasticsearch soft memlock unlimited`
	`elasticsearch hard memlock unlimited`

`bootstrap.memory_lock: true`	(Elasticsearch 5.6.x) Set to lock the memory on startup to ensure that Elasticsearch never swaps (swapping makes it perform poorly). Ensure that enough system memory resources are available for all processes running on the server.
	To allow the `elasticsearch` user to disable swapping and to increase the number of open file descriptors, the RPM installer makes these edits to`/etc/security/limits.d/10-caringo-elasticsearch.conf`:
	`# Custom for Caringo Swarm`
	`elasticsearch soft nofile 65536`
	`elasticsearch hard nofile 65536`
	`elasticsearch soft nproc 4096`
	`elasticsearch hard nproc 4096`
	`# allow user 'elasticsearch' memlock`
	`elasticsearch soft memlock unlimited`
	`elasticsearch hard memlock unlimited`

`threadpool.bulk.queue_size: 1000`	(Elasticsearch 2.3.3 only) Add to increase the indexing bulk queue size to compensate for bursts of high indexing activity that can exceed Elasticsearch’s rate of indexing.

`script.inline: true`
`script.indexed: true`
	(Elasticsearch 2.3.3 SwarmNFS users only) Add to support dynamic scripting.

`script.inline: true`	(Elasticsearch 5.6.x SwarmNFS users only) Add to support dynamic scripting.

`http.cors.enabled: true`
`http.cors.allow-origin: "*"`
	Add to support metrics in the Swarm Storage UI.

`path.data: <path·to·data·directory>`	By default, path.data goes to `/var/lib/elasticsearch` with the needed ownership. If you want to move the Elasticsearch data directory, choose a separate, dedicated partition of ample size, and be sure to make the `elasticsearch` user the owner of that directory: `chown -R elasticsearch:elasticsearch <path·to·data·directory>`

Systemd (RHEL/CentOS 7)

Create a systemd override file for the Elasticsearch service to set the LimitMEMLOCK property to be unlimited.

1. Create the override file:

```
/etc/systemd/system/elasticsearch.service.d/override.conf
```

2. Add this content:

```
[Service]
LimitMEMLOCK=infinity
```

3. Load the override file (otherwise, the setting will not take effect until the next reboot):

```
sudo systemctl daemon-reload
```

Environment Settings

Edit the environmental settings: `/etc/sysconfig/elasticsearch`

Elasticsearch 5.6.x

MAX_OPEN_FILES	Set to `65536`
MAX_LOCKED_MEMORY	Set to `unlimited` (prevents swapping)

Elasticsearch 2.3.3

MAX_OPEN_FILES	Set to `65535`
MAX_LOCKED_MEMORY	Set to `unlimited` (prevents swapping)
ES_HEAP_SIZE	Set to half the physical memory on the machine, but not more than 31 GB.

JVM Options

Edit the JVM settings: `/etc/elasticsearch/jvm.options`

Elasticsearch 5.6.x

-Xms	Set to half the available memory, but not more than 31 GB.
-Xmx	Set to half the available memory, but not more than 31 GB.

Log Setup for 5.6

To customize the logging format and behavior, adjust its configuration file: `/etc/elasticsearch/log4j2.properties`

1. In its default location, logging has the needed ownership. However, if you want to move the log directory, choose a separate, dedicated partition of ample size, and make the `elasticsearch` user the owner of that directory:

   ```
   chown -R elasticsearch:elasticsearch <path·to·log·directory>
   ```

2. Best practice - For better archiving and compression than the built-in log4j, turn off the rotation of log4j and use logrotate.
   a. Edit the log4j2.properties to limit the amount of space consumed by Elasticsearch log files in the event of an extremely high rate of error logging.
      Locate the file: section and make these edits:

Update to these values

```
appender.rolling.filePattern =
${sys:es.logs.base_path}${sys:file.separator}${sys:es.logs.cluster_name}-%i.log.gz
appender.rolling.policies.type = Policies
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = 2097152
```

Add these settings

```
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.max = 25
```

b. Adjust the log size and log file count for the deprecation log:

Update to these values

```
appender.deprecation_rolling.policies.size.size = 2097152
appender.deprecation_rolling.strategy.max = 25
```

c. Repeat the edits for the slowlog files:

Update to these values

```
appender.index_search_slowlog_rolling.policies.size.type =
SizeBasedTriggeringPolicy
appender.index_search_slowlog_rolling.policies.size.size =
2097152
...
appender.index_indexing_slowlog_rolling.filePattern =
${sys:es.logs.base_path}${sys:file.separator}${sys:es.logs.clus
ter_name}_index_indexing_slowlog-%i.log.gz
appender.index_indexing_slowlog_rolling.policies.type =
Policies
appender.index_indexing_slowlog_rolling.policies.size.type =
SizeBasedTriggeringPolicy
appender.index_indexing_slowlog_rolling.policies.size.size =
2097152
```

Add these settings

```
appender.index_search_slowlog_rolling.strategy.type =
DefaultRolloverStrategy
appender.index_search_slowlog_rolling.strategy.max = 25
...
appender.index_indexing_slowlog_rolling.strategy.type =
DefaultRolloverStrategy
appender.index_indexing_slowlog_rolling.strategy.max = 25
```

Log Setup for 2.3.3

To customize the logging format and behavior, adjust its configuration file: `/etc/elasticsearch/logging.yml`

1. In its default location, logging has the needed ownership. However, if you want to move the log directory, choose a separate, dedicated partition of ample size, and make the `elasticsearch` user the owner of that directory:

   ```
   chown -R elasticsearch:elasticsearch <path·to·log·directory>
   ```

2. Best practice - For better archiving and compression than the built-in log4j, turn off the rotation of log4j and use logrotate.

a. Edit the `logging.yml` to limit the amount of space consumed by Elasticsearch log files in the event of an extremely high rate of error logging.
Locate the `file:` section and make these changes:

Before

```
file:
    type: dailyRollingFile
    file: ${path.logs}/${cluster.name}.log
    datePattern: "'.'yyyy-MM-dd"
...
```

After

```
file:
    type: rollingFile                       # change from
dailyRollingFile
    maxBackupIndex: 0
    maxFileSize: 1000000000                 # 1 GB
    file: ${path.logs}/${cluster.name}.log
    # datePattern: "'.'yyyy-MM-dd"          # remove
...
```

b. Repeat for the deprecation and slowlog log files, as appropriate:

```
deprecation_log_file:
    type: rollingFile
    file: ${path.logs}/${cluster.name}_deprecation.log
    layout:
      type: pattern
      conversionPattern: "[%d{ISO8601}][%-5p][%-25c] %m%n"
    maxBackupIndex: 0
    maxFileSize: 1000000000 # (1GB)

  index_search_slow_log_file:
    type: rollingFile
    file: ${path.logs}/${cluster.name}_index_search_slowlog.log
    layout:
      type: pattern
      conversionPattern: "[%d{ISO8601}][%-5p][%-25c] %m%n"
    maxBackupIndex: 0
    maxFileSize: 1000000000 # (1GB)

  index_indexing_slow_log_file:
    type: rollingFile
    file:
${path.logs}/${cluster.name}_index_indexing_slowlog.log
    layout:
      type: pattern
      conversionPattern: "[%d{ISO8601}][%-5p][%-25c] %m%n"
    maxBackupIndex: 0
    maxFileSize: 1000000000 # (1GB)
```

Log Rotation

Finally, add a script to manage the log rotation. These are sample contents of a `logrotate.d` script (default location: `/etc/logrotate.d/elasticsearch`):

```
/var/log/elasticsearch/*.log
{
        weekly
        rotate 8
        size 512M
        compress
        missingok
        copytruncate
}
```

At this point, custom configuration is complete. Resume your Elasticsearch installation:

- Installing Elasticsearch

Installing Swarm Metrics

> For description of Swarm Metrics, its components, templates, and the set of rolling indices it generates, see Swarm Historical Metrics.

Index defaults

For Elasticsearch 5.6.x, Swarm Metrics indices are configured with 1 shard and one additional replica per index; for ES 2.3.3, indices are configured with 3 shards and one additional replica per index.

If more shards or replicas are desired, contact Support for help with that configuration.

Once the current version of Elasticsearch is running, install Swarm Metrics on one of the Elasticsearch servers or another system running RHEL/CentOS 7 or 6.

1. Install the new ES curator package. (See Elasticsearch for details.)
    a. Download and install the public signing key for Elasticsearch.

    ```
    rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch
    ```

    b. Create a yum repository entry. In `/etc/yum.repos.d`, create the file, "`curator.repo`" and include the section correct for your version of RHEL/CentOS:

RHEL/CentOS 7

```
[curator-5]
name=CentOS/RHEL 7 repository for Elasticsearch Curator 5.x
packages
baseurl=https://packages.elastic.co/curator/5/centos/7
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

RHEL/CentOS 6

```
[curator-5]
name=CentOS/RHEL 6 repository for Elasticsearch Curator 5.x
packages
baseurl=https://packages.elastic.co/curator/5/centos/6
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

c. Install the ES curator package.

```
yum install elasticsearch-curator
```

2. In the Swarm bundle, locate the Metrics RPM: `caringo-elasticsearch-metrics-<version>.noarch.rpm`
3. Install the metrics package.

```
yum install caringo-elasticsearch-metrics-<version>.noarch.rpm
```

Upgrade error

If the elasticsearch-curator package shows an error during an upgrade, this is a known curator issue. The workaround is to reinstall the curator:

```
yum reinstall elasticsearch-curator
```

4. Update the configuration file for the metrics curator: `/etc/caringo-elasticsearch-metrics/metrics.cfg`

clusters	Set to one or more clusters for which you want metrics collected. Use spaces or commas to separate multiple values. `cluster1.example.com cluster2.example.com`
hosts	Set to one or more Elasticsearch servers (bound name or IP address) to which you want to publish metrics. Use spaces or commas to separate multiple values. `es1.example.com es2.example.com` Metrics requirement If you configured the Elasticsearch host (in `elasticsearch.yml`) to a specific hostname or IP address, then the host for Metrics must match. However, if you configure `network.host` in `elasticsearch.yml` to be "`_site_`" (recommended), then the host in `metrics.cfg` can be a valid IP address or hostname for that Elasticsearch server.
<all else>	Accept or modify the remaining configuration values.

5. Configure the metrics settings for the Swarm Storage cluster.
Using either the Swarm UI or SNMP, update these settings. (See Persisted Settings for how to update settings via SNMP.) Note that the dynamic (SNMP-enabled) values in the config file only affect a new cluster on first deployment.

Swarm Storage Setting	SNMP Name	Default	Description
metrics.target metrics.targets	metricsTargetHost	none (disabled)	Required. One or more Elasticsearch servers (a fully qualified domain name or IP address) where metrics-related statistics are captured. Use spaces or commas to separate multiple values. To disable statistics collection, leave the value blank.
metrics.port	metricsTargetPort	9200	The port on the Elasticsearch server where metrics-related statistics are captured.

metrics.period	metricsPeriod	900	In seconds, from 15 seconds to 1 day; defaults to 15 minutes. How frequently to capture metrics-related statistics.
metrics.diskUtilizationThreshold		5	In percent, from 0 to 100; defaults to 5 percent. Minimum percentage of Elasticsearch disk space available before metrics stop being indexed. Indexing resumes when space is greater than this minimum. (v9.1)
metrics.diskUtilizationCheckInterval		600	In seconds, from 15 seconds to 1 day; defaults to 10 minutes. How frequently to check disk utilization on the Elasticsearch cluster. (v9.1)

6. To start collecting metrics, manually run curator to prime the indexing setup, which defines the metrics schemas, creates empty indices with those schemas, and sets up aliases to the indices. (By default, the curator runs at midnight; however, for a new installation, it will run at the top of the next hour.) Running the curator ensures that the current day's indices exist and that all the aliases are up to date, so that metrics can begin to be collected (that is, priming does not generate any metrics data).
   To run curator manually, use this command:

   ```
   /usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n
   ```

   Tip
   The Curator checks for an existing search index to validate the cluster name. If you have a new cluster or are not using a search feed, add -v (--valid) to skip the check. (v9.4)

   After metrics has been configured in the Swarm cluster, the first metrics data will appear within the number of seconds of `metrics.period`, which defaults to 15 minutes.

   Caution
   The Swarm UI and legacy Admin Console will log CRITICAL messages if metrics is misconfigured or if connection to the Elasticsearch cluster is lost.

7. Verify that metrics is configured correctly by running this command:

```
curl
"http://ESHOST:9200/metrics-CLUSTERNAME-scsp-all/metrics/_search?pr
etty=true"
```

If needed, see Swarm Metrics Troubleshooting.

8. Resume your Elasticsearch installation: Installing Elasticsearch

Swarm Storage UI Installation

This section describes how to install the Swarm Storage UI. To use the legacy Admin Console (`http://{cluster}:90`), no additional installation is required.

> See also these sections:
>
> - Storage UI 1 Release Notes
> - Swarm Storage UI (usage guide)

System Requirements

Browser	The Storage UI requires a JavaScript-enabled internet browser. Development and testing were conducted using the most recent versions of Firefox and Chrome.
Network	The system where the Storage UI is installed must have direct access to the Swarm Storage VLAN. Adjust firewall rules to allow the direct access required: • Swarm Storage nodes (port 91 by default) • Elasticsearch nodes (port 9200 by default).

Dependent Software	Swarm Storage version 9.2 or later, with the following configuration to enable the trend charts to display data:
	• A Metrics instance, configured and available.
	• Elasticsearch must be configured to work with web applications with the following settings:
	`http.cors.enabled: true`
	`http.cors.allow-origin: "*"`

Installing the Storage UI

The Swarm Management UI supports different installation methods, depending on where it will be installed. If you have a Content Gateway implemented, the best practice is to install it there.

> Note
> The Management UI connects to port 91. If you change the bind port to anything other than 91, users will have to specify it when logging in: `{host}:{custom-port}`

Installing on Content Gateway

The Storage UI will easily install on a server running the Content Gateway, and it is the recommended implementation. To add the UI to a server running Gateway:

1. Locate the Storage UI RPM in the Swarm 9 distribution package.
2. Copy the UI rpm to the directory where the Gateway was installed, and run the following:

```
yum install caringo-storage-webui-{version}.rpm
```

3. To access the UI, browse to:

```
http://{hostname}/_admin/storage
```

Installing on CSN 8.3

> See Scenarios for Storage UI for how to add Storage UI under different CSN usage scenarios.

To use the Swarm Storage UI with Swarm Metrics, you need all of the enabling components, such as a simplified Gateway (Service Proxy) to provide the needed access and an Elasticsearch service and curator. Your CSN download includes all of the RPMs and scripts needed to perform the installation and configuration of these components.

Caution

- Do not using the CSN-embedded ES server as a Swarm Search Feed target.
- Do not change the preconfigured defaults for Metrics, which are set in `/etc/caringo-elasticsearch-metrics/metrics.cfg`
  - `metrics.target` and `metrics.port` are preconfigured for the Service Proxy.
  - Metrics are checked every 15 minutes and are retained for 120 days.
- If you have any scaling concerns, consult Support about whether you need a dedicated Elasticsearch cluster.

1. Install the CSN bundle as usual (the new RPMs will install for you).
2. Add a Swarm license and start up the Storage nodes.
3. Run the integration script to configure the Service Gateway, adding arguments as appropriate:

```
/usr/bin/configure_storage_webui_with_serviceproxy.py
```

Optional arguments	Default	Purpose
-h, --help		Display help summary and then exit.
-v, --verbose	INFO	Add to generate more extensive logging output.
-d DIR, --directory DIR	/var/log/caringo	Specify a different working directory for logs and output.
-s, --start_services	yes	Disable automatic enabling and start up of services after configuration.
-n NUM_HOSTS, --num_hosts NUM_HOSTS	5	Specify a different number of hosts to add to the Gateway configuration.
--indexer_host INDEXER_HOST [INDEXER_HOST ...]		Specify remote Elasticsearch hostnames or IPs, disabling (and preventing creation of) a local ES instance. Important: If you have an existing ES cluster, be sure to specify it here to avoid creating an extra local instance.

4. Each time the script prompts you for a configuration setting, press Enter to accept the suggested value, or enter your own.
5. On completion, the script will attempt to start all of the services with their new configurations, logging the output to the `integration.log` file (which defaults to `/var/log/caringo`).
6. When the script provides you with your URL and credentials, log into the Storage UI with a JavaScript-enabled browser (development and testing were conducted using the most recent versions of Firefox and Chrome).

Note

> Metrics data will not appear until new Storage data is added and metrics collection cycles have occurred, which is a minimum of 30 minutes.

7. The script creates a "`caringoadmin`" user (password "`caringo`"), which is a user that is local to the CSN. These user credentials are required to login to the Storage UI through the Service Gateway.

   > Note
   > This `caringoadmin` user is only used by the Service Gateway, not by the Swarm cluster.

   a. To change the `caringoadmin` password on the local CSN, run the password command and follow the prompts:

   ```
   passwd caringoadmin
   ```

   b. The user `caringoadmin` user is defined as the administrator for the Service Gateway in this file on the CSN: `/etc/caringo/cloudgateway/policy.json`. The user can be changed according to your requirements, just as in the Content Gateway.
   See Gateway Access Control Policies.
8. With CSN 8.3, all of these interfaces remain available and may be used concurrently:

Legacy CSN Console	`http://{CSN·host}:8090`
Legacy Admin Console	`http://{CSN·host}:8090/services/storage`
Swarm Storage UI	`http://{CSN·host}/_admin/storage`

Upgrading the Storage UI

To upgrade the Storage UI, follow this sequence according to the guidance in the release notes:

1. As needed for this release, complete any upgrades of Swarm components in this order:
   a. CSN Platform Server
   b. Swarm Storage
   c. Elasticsearch (for S3 and Swarm Historical Metrics) — Create a new feed, let it finish, then make it Primary.
   d. Content Gateway
2. Copy the UI rpm to the directory where the Gateway was installed, and run the following:

```
yum install caringo-storage-webui-{version}.rpm
```

> Note

> If upgrading from a version prior to 1.1.0 (`caringo-storage-webui-1.1.0-1.noarch.rpm`), use the command `yum upgrade` instead of `yum install`.

Content Gateway Implementation

> See also these sections:
>
> - Content Gateway Release Notes
> - Swarm Content Gateway (usage guide)
> - Content Application Development
> - S3 Protocol Interface

System Requirements

This section covers the hardware requirements and software installation and configuration of the Gateway platform components, for those in the following roles who need to deploy and manage the Gateway:

- Storage system administrators
- Network administrators
- Technical architects

The administrators are normally responsible for allocating storage, managing capacity, monitoring storage system health, replacing malfunctioning hardware, and adding additional capacity when needed. Network administrators are responsible for TCP/IP switching, routing, load balancing, and firewall setup. Herein, these different roles will be referred to simply as administrators.

> Note
> This section assumes that you are familiar with the deployment and management processes for Swarm Administration and have knowledge of TCP/IP networking, basic x86 hardware setup, and intermediate Linux system administration.

- Gateway Requirements
- Configuring Swarm Storage for Gateway
- Gateway Installation
- Gateway Configuration

Gateway Requirements

- Prerequisites
- System Requirements
- Space Requirements
  - Spool space for multipart uploads
  - Logging space
- S3 Requirements

The Gateway service is essentially a reverse proxy with some protocol inspection duties. As a proxy between the client applications and the storage nodes, its primary duty is to pass bytes from one network adapter to another.

Prerequisites

Content Gateway requires the following components for installation and operation:

- Swarm Storage cluster implemented with Storage settings needed by Gateway
- Elasticsearch cluster installed

> Important
> If you will be using the usage metering data for critical functions such as billing, be sure to deploy at least three Elasticsearch servers (for high availability) and to implement snapshot backups of that data.

- Search feed defined and enabled
- Network time protocol (NTP) server reachable by both Gateway server(s) and Swarm storage nodes
- At least one server on which to install the Content Gateway software

System Requirements

The system requirements for the Gateway depend on the volume of the traffic and the speed of the upstream network connection to your client applications.

- Gateway server software:
  - 64-bit Linux operating system (RHEL/CentOS 7 is recommended)
  - Java 8 (earlier versions of Java are not supported)
- Gateway server hardware:
  - Virtual or physical machine
  - 2+ CPU cores
  - 2+ GB RAM
  - 3+ GB `/tmp` space
  - 2+ GB available disk storage after OS installation (see Space Requirements below)
- For high availability and capacity scaling, add the following:
  - Two or more additional Gateways
  - A load-balancing mechanism
- Prevent Gateway clients from making storage requests directly to the back-end storage cluster using one of these methods:
  - (most common) Make the Gateway servers dual-homed on the front-end client network and the back-end storage network.
  - Use network filtering to prevent direct user access to the storage cluster and to deploy Gateway servers and storage servers on one subnet.
  - Use VLAN tagging on the Gateway server's network interface to allow one physical interface to carry both front-end and back-end traffic.

Space Requirements

Spool space for multipart uploads

The HTTP multipart MIME upload operation requires spool space on the Gateway server; all other operations, including the S3 multipart upload, SCSP multipart writes, and normal whole-object writes, stream through the Gateway and directly to the back-end object storage nodes. HTTP multipart MIME POST requests are used by the upload function in the Content Portal and by HTML form POSTs.

Make sure that the total free disk space on a Gateway server includes an allowance for the maximum that you expect to be needed for these requests. To control the spool location and the percentage of disk space that can be used, set the `multipartSpoolDir` and `multipartUsageAllowed` in the `[gateway]` section of the configuration file.

> Best practice
> If your Content UI Overview has many users and/or large uploads, increase the available space in the Gateway's spool directory to 32 GB or more and increase the `multipartUsageAllowed` parameter value.

Logging space

In the default configuration, the Gateway server will use up to 2GB of disk space for application logs and audit logs. The retention time and file size of the historical logs can be changed as required based on your deployment requirements. See Logging Configuration in the Gateway configuration section.

S3 Requirements

To use S3 with Content Gateway, follow these requirements:

- Enable and configure erasure-coding (EC).
- Size your cluster to support EC; for example, do not attempt to use S3 with inadequate resources, such as 3 chassis and reps=2.

> See Configuring Swarm Storage for Gateway and also Erasure Coding EC and Hardware Setup in the Swarm Storage guide.

Configuring Swarm Storage for Gateway

This section provides information specific to running Swarm Storage with Gateway. First, you need to install and configure Swarm, the storage cluster (storage nodes are appliances that run on dedicated hardware).

Network Placement

When deployed with Gateway, the storage nodes should be placed on a network subnet that is not directly accessible to the client applications. This way, all user communications with the storage cluster must go through the Gateway.

> Caution
> If users are allowed to communicate directly with the storage cluster nodes, they may bypass access security, the business rules for content metadata, and audit logging that is performed by the Gateway and may render content in the cluster unusable to the Gateway. Only allow direct access to the storage cluster nodes under highly controlled circumstances, such as administrator-only operations or trusted applications.

Domain Management

The Swarm cluster provides for logical separation of content among multiple tenants through the use of storage domain names. Gateway has the following requirements above and beyond those for a baseline storage deployment and client usage.

- An administrative domain must be created in the storage cluster.
- At least one storage domain must be created in the storage cluster.
- Storage domains must adhere to IANA naming standards (that is, be valid DNS names).
- Client applications must specify a storage domain in every request.

The storage domain name for an operation is specified by the client application according to the following precedence from highest to lowest:

- SCSP `domain=X` query argument
- HTTP `X-Forwarded-Host` header
- HTTP `Host` header

In order to make use of the Host header to identify the storage domain with most HTTP/1.1 libraries, storage domains in Swarm must resolve to least one IP address ("A" record) for client applications. Additionally, the resolved IP address should be for a Gateway or, if applicable, some other front-end network appliance such as a load balancer. If there are multiple Gateway servers, using a DNS round-robin with their IP addresses is a valid configuration to use.

This is an example of a BIND 9 zone file that implements a wildcard of all storage domains within the `cloud.example.com` parent DNS domain and points them to the IP address `10.100.100.100`.

```
$TTL 600 @ IN SOA cloud.example.com. dnsadmin.example.com. (
    2016070201 ; Serial number
    4H     ; Refresh every 4 hours
    1H     ; Retry every hour
    2W     ; Expire after 2 weeks
    300 )  ; nxdomain negative cache time of 5 minutes
IN NS ns1.example.com.
* IN A 10.100.100.100
```

In the example zone file, `10.100.100.100` is the IP address used by client applications to communicate with the Gateway or a front-end load balancer. The names `hydrogen2.cloud.example.com` and `oxygen.cloud.example.com` would both resolve to the same IP address.

Elasticsearch Servers

When using the S3 storage protocol, the metadata search service must be accessible to the Gateway servers.

When deployed with Gateway, like the storage nodes, the typical placement will be on a network subnet that is not directly accessible to the client applications. At this time, there are no end-user supported API calls directly to the metadata search service.

Best practice

When defining the search feed for use with Gateway, set the batch timeout value to a low value (such as 1 to 5 seconds) in order to provide good interactive search-after-create response to the client applications that use the Gateway.

Configuration Requirements

When using Swarm Storage with Gateway, you must use these Swarm configuration settings and adhere to the following operational changes. These configuration changes refer to the configuration file(s) for Swarm. This is either the `node.cfg` file or, when using a Platform Server to manage the nodes, the global configuration file for the cluster:

```
/var/opt/caringo/netboot/content/cluster.cfg
```

Caution

Failure to use these settings and operational changes can prevent Gateway from working properly with the storage cluster.

Requirement	Description

Enable `ecEncoding`	You must enable and configure erasure coding; if not, the Gateway cannot start (`service cloudgateway init` will fail).
	`[policy]`
	`ecEncoding = {k:p}`
	`ecMinStreamSize = 1MB`
	See Implementing EC Encoding Policy.

Enable `enforceTenancy`	Enforce the use of tenancy for unnamed objects by setting:
	`[cluster]`
	`enforceTenancy = True`

<table>
<tr><td>Enable <code>noauth</code></td><td>Be sure to disable the legacy Swarm authentication/authorization by setting:<pre>
[security]
noauth = True
</pre></td></tr>
<tr><td>Storage Domain Management</td><td>Only create and manage storage domains through the Content Portal or programmatically through the Gateway's management API.

Important
When using Content Gateway, do not use the legacy Admin Console (port 90) to manage storage domains.

Troubleshooting: A domain created by the legacy Admin Console (port 90) contains the legacy C astor-Authorization header, which causes the Content UI to report "Page Not Found: The original bucket to which this collection refers cannot be found or has been replaced..." The fix is to remove the Castor-Authorization header (which removes Swarm access control and assumes your Swarm nodes are accessible only to trusted clients) and ensure that "caringoadmin" is a Gateway LDAP or PAM user (idsys.json). To remove the header, issue a COPY request directly to a Swarm node (the semicolon syntax to remove header values requires a curl version later than 7.29).<pre>
curl -v -u admin --post301 --location-trusted -XCOPY
    -H 'X-Owner-Meta: caringoadmin'
    -H "content-type:application/castorcontext"
    -H 'Castor-Authorization;'
    'http://SCSP_HOST/?domain=myolddomain&preserve&admin'
</pre></td></tr>
</table>

Gateway Installation

The Content Gateway is the access point and gatekeeper for the back-end storage cluster. It also provides value-added services for user applications and storage administrators.

> Important
> Installing a Content Gateway on a CSN is not recommended.

These installation instructions are relevant to RHEL/CentOS 6 and 7, which are the supported platform for the Gateway.

> Upgrading

> For information about upgrading, see the Upgrading section in the Release Notes for your version.

1. Apply all current operating system patches before you install the Gateway.

   > Note
   > The installer preserves any existing versions of `pip` and `requests` that it detects.

2. Install Java:

   > Required
   > Gateway requires a Java 8 JDK. The JDK must be Oracle's Java or the OpenJDK. Use the same JVM and version on all of your Gateways.

   a. Install the Java language on the server:

   ```
   yum install java-1.8.0-openjdk
   ```

   b. After installing, verify that the correct Java version is active:

   ```
   java -version
   ```

   c. If you need to change the active Java version, run the following command:

   ```
   alternatives --config java
   ```

3. From Caringo Connect (connect.caringo.com), download the Swarm 9 distribution package to get the Gateway distribution, and unzip it.
4. Locate the RPM for the Gateway software. If you have not previously added the Caringo RPM public key that is included with the distribution bundle to your system, run the following command to do so:

   ```
   rpm --import RPM-GPG-KEY
   ```

5. Run the following command to install the Gateway package, substituting the exact version number for the RPM in the distribution file for the {version} string:

```
yum install caringo-gateway-{version}.rpm
```

6. Go to the `examples` directory for configuration file examples to study and clone for your own use:

```
/etc/caringo/cloudgateway/examples
```

7. Complete the IDSYS document for user authentication:

```
/etc/caringo/cloudgateway/idsys.json
```

8. Complete the Policy document for access control:

```
/etc/caringo/cloudgateway/policy.json
```

9. Note the location for the server logs:

```
/var/log/caringo/cloudgateway_server.log
```

10. Verify that NTP time synchronization is being used on the Gateway server to ensure proper storage transaction handling and that the audit log time stamps match across servers. NTP is critical for the operation of Swarm and should be used on all hosts that interact with Swarm.

Gateway Configuration

- Configuring the Content Gateway
  - Minimum Configuration
  - Configuration Sections
  - Administrative Domain
- Enabling the Service Proxy
- Configuring Logging
  - System Logging
  - Audit Logging

After installing the Content Gateway service, these configuration files will be on your system:

```
/etc/caringo/cloudgateway/gateway.cfg
/etc/caringo/cloudgateway/logging.cfg
```

Configuring the Content Gateway

Minimum Configuration

While cluster administrators must understand the details of configuring Content Gateway, this section summarizes the minimum steps required to configure and run Gateway. To deploy Gateway into production, additional customization is needed.

1. Edit the `/etc/caringo/cloudgateway/gateway.cfg` file:
   a. Set the `adminDomain` parameter to the name of an administrative domain that will be created.
   b. Set the `hosts` parameter for your storage cluster nodes. Including 3 to 5 nodes is sufficient for most deployments.
   c. Include the ES server in the "server" line, without which Gateway cannot start.
   d. Enable at least one of the front-end protocols: SCSP or S3.
   Alternatively, for Service Proxy only use (to host the Swarm UI), set both to disabled and complete the `[cluster_admin]` section.
2. Create the administrative domain by running:

   ```
   /opt/caringo/cloudgateway/bin/initgateway
   ```

3. Start the Gateway service:

   RHEL/CentOS 6

   ```
   service cloudgateway start
   ```

   RHEL/CentOS 7

   ```
   systemctl start cloudgateway
   ```

4. Enable automatic startup of the Gateway service

RHEL/CentOS 6

```
chkconfig cloudgateway on
```

RHEL/CentOS 7

```
systemctl enable cloudgateway
```

5. Check either that `IPTABLES` are off or that inbound access for the front-end protocols is allowed. These commands will turn off and disable the firewall daemon.

RHEL/CentOS 6

```
chkconfig iptables off
service iptables stop
```

RHEL/CentOS 7

```
systemctl disable firewalld
systemctl stop firewalld
```

6. Follow this guide for configuring and running the Gateway service. Production deployments will require customizations of the configuration parameters, below.

Configuration Sections

The `gateway.cfg` file controls the core operations of the Content Gateway. It is a plain text, INI-formatted file that is read when the Gateway is first started. The parameters within the file are organized into the following sections.

> Docker or proxy use
>
> As of release 5.4, you can configure Gateway to be used either within a Docker environment or behind a proxy. The configuration has two new settings available (`externalHTTPPort, externalHTTPSPort`) per protocol: [scsp] and [cluster_admin], the Service Proxy. They only take effect when X-Forwarded-Proto is found on the request; Gateway uses X-Forwarded-Proto to determine which port to use.

- [gateway]
- [storage_cluster]
- [scsp]
- [s3]
- [metering]
- [caching]
- [quota]

[gateway]

This section configures client communications:

adminDomain		The administrative domain where meta information about tenants and storage domains is kept. This parameter is required and must be set to the same value for all Gateway servers. This should not match the Swarm default domain (cluster.name).
threads	Defaults to 100 times number of CPU cores Minimum of 200	The number of threads allocated to handling client requests. For CPUs with hyperthreading enabled, this calculation is based on the number of virtual cores, not physical.
tokenTTLHours	Defaults to 24	The default number of hours an authentication token is valid if no time is defined when it is created.
multipartSpoolDir	Defaults to `/var/spool/cloudgateway`	The location of the spool directory for HTTP multipart MIME upload temporary space. These requests are made by the upload page within Content Portal.
multipartUsageAllowed	Defaults to 50	The percentage of the file system that can be used for multipart MIME upload temporary space.
recursiveDeleteMaxThreads	Defaults to 50	The maximum number of parallel delete operations to dispatch when processing recursive delete requests.
sanitizeErrors	Defaults to false	Set to true in order to hide identity management configuration details from authentication errors.

cookieDomains		One or more base domains for the `Set-Cookie` response header to scope (instead of the FQDN from the request) if an authentication token is created within a child domain of one of these base domains. This can be useful when using the Content Portal to access multiple storage domains that share a common base domain when you want to use the same authentication token across the domains. (v5.2.2) Example: `cookieDomains = cloud.example.com cloud.example.net`

[storage_cluster]

This section configures the back-end storage cluster:

locatorType		Set to "static" or "zeroconf". If not present, and `clusterName` is set, Zeroconf is assumed. Otherwise, an error is raised and the service will not start. When using Zeroconf on a multi-homed Gateway with more than one network interface, `clientBindAddress` must be defined.
hosts		Space-delimited list of IP addresses or host names of the storage cluster nodes. Required for static locator type.
port	Defaults to 80	Integer socket port number for SCSP on the storage nodes.
clusterName		The name of the storage cluster. Required for zeroconf locator type, otherwise ignored.
indexerHosts		Space-delimited list of the Elasticsearch metadata index servers used by the storage cluster. Required for the S3 protocol and for Content Metering. Note: If the Primary Swarm Search feed changes, be sure to verify these settings and restart the Gateway servers.
indexerPort	Defaults to 9200	The socket port on which the Elasticsearch servers listen.
clientBindAddress	Defaults to 0.0.0.0	Set to the IP address of the network interface connected to the storage cluster subnet when using a multi-homed Gateway. The value must be defined as a non-default value when using a multi-homed Gateway server such as one that is connected to a front-end client network and a back-end storage network.

maxConnectionsPerRoute	Defaults to 100	The maximum number of open connections to a specific storage node.
maxConnections	Defaults to 250	The maximum number of open connections to allow. This includes both active and idle connections.
connectTimeout	Defaults to 60	The time in seconds allowed to connect to a node.
socketTimeout	Defaults to 120	The time in seconds allowed for an active connection to deliver data.
idleTimeout	Defaults to 120	The time in seconds that an idle socket is allowed to remain in the connection pool.
continueWaitTimeout	Defaults to 30	The time in seconds to wait for client response after a 100 continue reply.
dataProtection	Defaults to "immediate"	Controls whether synchronous (immediate) or asynchronous (delayed) data protection is requested when writing to the storage cluster. Values are: "immediate", "delayed". Important To disable ROW (replicate on write) behavior, you must set `scsp.replicateOnWrite=false` in the storage cluster in addition to setting `dataProtection = delayed`.
blockUndeletableWrites	Defaults to true	When enabled, the Gateway rejects any SCSP write (PUT, POST, COPY, APPEND) that includes a `deletable=no/false` lifepoint. This restriction applies to both named and unnamed (alias and immutable) objects. The request is refused with a 400 error message, "Unable to write undeletable object".

[scsp]

This section configures the front-end SCSP protocol. This protocol must be enabled for any Gateway that services Content Portal requests.

enabled	Defaults to true	Activates this protocol: Values are: "true", "false".

bindAddress	Defaults to 0.0.0.0 (all interfaces)	The IP address of the network interface to which the listening socket should bind.
bindPort	Defaults to 80	Integer socket port number for protocol; must be unique from S3 port if both enabled.
externalHTTPPort externalHTTPSPort		Optional, one or both. Allows Gateway to be used either behind a proxy or within a Docker environment, only taking effect when X-Forwarded-Proto is found on the request. (Gateway uses X-Forwarded-Proto to determine which port to use.) (v5.4)
allowSwarmAdminIP	Defaults to undefined	Allows the use of internal Swarm requests for content replication to pass through the Gateway. This is useful if you are using replication feeds between clusters that use Gateway as their front-end. Values are "all", full IP addresses, IP address prefixes, or a list of IPs/prefixes. When undefined, no addresses are allowed to send Swarm admin requests through the Gateway.

[s3]

This section configures the front-end S3 protocol, which is optional.

enabled	Defaults to false	The protocol must be explicitly enabled.
bindAddress	Defaults to 0.0.0.0 (all interfaces)	The IP address of the network interface to which the listening socket should bind.
bindPort	Defaults to 80	Integer socket port number for protocol; must be unique from SCSP port if both enabled.

enhancedListingConsistency	Defaults to true	Improves compatibility with S3 clients and software libraries that expect consistent listings (despite the documented nature of listings to be eventually consistent). Can be disabled to boost write throughput (especially for small objects), if listing consistency is not critical. (v5.2.1) Exceptions to synchronous indexing: • Deletes of manifests for canceled multipart uploads are done asynchronously. • On a delete, when there is not enough space on the local node to write a delete marker for a named object, Swarm writes to another node and indexes asynchronously. • On a rename, Swarm indexes the new name synchronously, but the old name is deleted asynchronously. • On a parallel write complete, the init stream is deleted asynchronously.

[metering]

This section configures usage metering, which is optional. See Content Metering.

enabled	Defaults to false	The feature must be explicitly enabled.
flushIntervalSeconds	Defaults to 300 (5 minutes)	How frequently to send usage reports to Elasticsearch. Minimum is 10 seconds. The default value is optimized for the resolution of the queries.
retentionDays	Defaults to 100 (days)	How long to retain usage records. Minimum is 2 days. If you significantly increase the retention period, allow for additional storage space.
storageSampleIntervalSeconds	Defaults to 3600 (1 hour)	How frequently to sample the disk usage. Minimum is 900 (15 minutes). Larger values reduce the query workload on Elasticsearch.

[caching]

This section configures cache expiration. Times are in seconds. To disable, set to 0.

authRefresh	Defaults to 300	Time before authorization is revalidated with a request to the identity management system.
tokenRefresh	Defaults to 300	Time before an authentication token is revalidated with a request to the administration domain.

idsysRefresh	Defaults to 300	Time that an IDSYS document is cached in memory.
policyRefresh	Defaults to 300	Time that a tenant, domain, or bucket Policy document is cached in memory.
xformRefresh	Defaults to 300	Time that an XFORM document is cached in memory.
metadataRefresh	Defaults to 300	Time that metadata for a tenant, domain, or bucket is cached in memory. This includes the owner for a tenant/domain/bucket and whether a bucket exists.
domainExistenceRefresh	Defaults to 300	Time that the knowledge of a domain's existence or nonexistence is cached.

[quota]

This section configures storage and network usage quotas. See Setting Quotas.

When enabled, the Gateway regularly refreshes its cache of quota information via an Elasticsearch query against usage metrics; if any quota limit is reached, it changes the quota state and performs the action specified by policy.

enabled	Defaults to false	The feature must be explicitly enabled.
minRefreshDeadline	Defaults to 60	The global limits on the speed of quota data refreshing. To increase the precision of the usage data, lower these values. To reduce the load on Elasticsearch, increase these values. To optimize the load on Elasticsearch, Gateway refreshes with a dynamic algorithm: slower when metrics are still far from the limit and faster when the limit approaches, slower when approaching a limit and faster as the overage nears an end. The minimum and maximum deadlines refer to the caps to apply to this refresh rate (no faster and no slower than these values).
maxRefreshDeadline	Defaults to 3600	
numRefreshThreads	Defaults to 4	The number of threads in the pool that continuously look at the most urgent deadlines in the queue and perform the refreshes (Elasticsearch queries) as needed.
maxRefreshRetries	Defaults to 3	The number of times that a refresh can fail due to a failing Elasticsearch query before an error is logged and the refresh is dropped.
maxQueueSize	Defaults to 10000	Maximum queue size for scope quota evaluations. The internal implementation uses a deadline queue and, If the queue is overflowed, the least urgent items will be pushed out of the queue.

queryTTL	Defaults to maxRefreshDeadline	This avoids unnecessary load on Elasticsearch by allowing the results of a quota check that is done when a scope (tenant, domain, bucket) is accessed to be cached for this period of time. If the time since last access is less that this value, the scope will not be scanned in the background. Setting this parameter to 0 disables the access caching function.
refreshRetryDelay	Defaults to 10	Number of seconds to wait before retrying a refresh after the previous failed due to a failing Elasticsearch query.
refreshIdleSleep	Defaults to 3	Seconds to wait after finishing the work in a queue and before starting again.
smtpHost	Defaults to blank	Required. The hostname or IP address of the SMTP server that will send the email notifications.
smtpPort	Defaults to 25	Optional. The port where the SMTP server listens.
smtpUser	Defaults to blank	Optional. The user to authenticate with SMTP server.
smtpPassword	Defaults to blank	Required if smtpUser is specified. The user password.
mailFrom	Defaults to `donotreply@caringo.com`	Email address for the sender of the notification.
mailSubjectTemplate	Defaults to `Quota state change notification`	Email templates for subject line and body. These variables can be used in both the subject line and message body templates. ● %metric% ● %state% ● %contextType% ● %contextName% The %xxx% strings render current values when the message is generated.
mailTemplate	Defaults to `Metric %metric% changed to %state% state in %contextType% %contextName%`.	

> See Setting Quotas.

Administrative Domain

Content Gateway uses one storage domain within the storage cluster in order to persist meta information about all tenants and storage domains. Although there is no difference between storage domains to the storage cluster, Content Gateway uses these two distinctions for domains: administrative domain, tenant storage domain.

- administrative domain refers to the domain used by Gateway in order to store meta information used in the management of tenants and all other storage domains, including itself, and should only be accessible to cluster administrators. While the administrative domain can be used to store general-purpose content, this is not recommended since care must be taken not to interfere with the objects managed by the Gateway.
- tenant storage domain (or just storage domain) refers to the domains that store content that is accessible to normal users and applications. All content within a tenant storage domain is potentially accessible to the users

of that domain and there is no special Gateway content within it.

The requirements for the name of the administrative domain are that it must be:

- globally unique for a set of tenant storage domains
- defined in the `gateway.cfg` file
- created prior to using tenant storage domains
- same for all Gateway servers servicing a set of tenant storage domains

> Important
>
> The content within the administrative domain must be protected from access by users other than the cluster administrators. Thus, when this domain is created, an owner must be set and, optionally, an appropriate domain Policy should be defined for it.

To facilitate the setup of the administrative domain, Gateway includes a command to properly create a locked-down domain. In order to use the command, edit the gateway.cfg file's adminDomain parameter, define the name for the administrative domain, and then run the command:

```
/opt/caringo/cloudgateway/bin/initgateway
```

> Caution
>
> This command should be run only one time when installing the first Gateway server; it should not be run when installing subsequent servers. Do not, under any circumstances, run it in a remote cluster to which you will replicate the administrative domain via a Feed.

A domain named by the `adminDomain` parameter will be created in the storage cluster with the owner set to the value `admin@`. Without additional action on the part of the cluster administrator, this domain is locked for all access and requires the use of an administrative override in order to log into the domain.

> See Restricting Domain Access for more information about access control and administrative override.

If cluster administrators want to open the access of the administrative domain, they can use the Policy and IDSYS documents for the domain and change the ownership by modifying the `X-Owner-Meta` metadata value.

> Caution
>
> Take care if access to the administrative domain is unlocked. Content stored within the administrative domain controls access, policies, and management data for all tenants and storage domains.

The name of the administrative domain must be unique for a set of tenant storage domains and must not be created more than once whether using an SCSP operation or by using the `initgateway` script. Once an administrative domain or a tenant storage domain has been created, the only proper way to instantiate the domain in another cluster is by using remote replication in Swarm.

> See Replicating Domains to Other Clusters.

Enabling the Service Proxy

On the Gateway instance that will run the Service Proxy, make the following changes to its configuration (`gateway.cfg` file):

> Important
> All of these settings are required. If your existing configuration file does not include the `[cluster_admin]` section, be sure to add it

[cluster_admin]	`enabled=true`	Enables the Service Proxy functionality.
	`bindAddress=<IP|hostname>`	Specifies the IP address or host name where Service Proxy listens for incoming storage cluster management API and Metering Query requests.
	`bindPort=91`	Specifies the port where Service Proxy listens. By convention, this is port 91.

	`externalHTTPPort=<port>`
	`externalHTTPSPort=<port>`
		Optional, one or both. Allows Gateway to be used either behind a proxy or within a Docker environment, only taking effect when X-Forwarded-Proto is found on the request. (Gateway uses X-Forwarded-Proto to determine which port to use.) (v5.4)

	`secretKey=<key>`	If you have multiple gateways (running on dual CSN's for HA / load balancing, be sure to set secretKey so that the Service Proxy's link obfuscation is consistent across gateways. Otherwise a random key will be generated whenever gateway is started, causing open browsers to break (screen fills with red boxes) until the browser itself is closed and reopened.
	`testMode=<true|false>`	When troubleshooting, you can enable testMode, which stops obfuscation of the backend Swarm Storage and Elasticsearch node IPs.
[storage_cluster]	`managementPort=91`	Specifies the port where Swarm listens for storage cluster management API requests. By convention, this is port 91.

	`managementUser=<Swarm·admin·user>`	Specifies the user known to Swarm that is allowed to perform management API requests against the storage cluster.
	`managementPassword=<Swarm·admin·password>`	Specifies the password of the `managementUser`.
[s3]	`enabled=false`	Important: When using Gateway just as a Service Proxy, be sure to retain the [s3] and [scsp] sections and explicitly add '`enabled=false`'.
[scsp]	`enabled=false`	

Authentication and authorization for the Service Proxy uses Content Gateway's root IDSYS and root Policy. The root Policy must grant all actions to the storage administrator users and/or groups. For example:

```
{
    "Version": "2008-10-17",
    "Statement": [{
        "Action": ["*"],
        "Resource": "*",
        "Effect": "Allow",
        "Principal": {"user": ["admin", "admin2"]},
        "Sid": "storage-admins"
    }],
    "Id": "id-170428899"
}
```

Configuring Logging

The `logging.cfg` file is a standard log4j configuration file. This section describes how to configure system and audit logging for the Gateway service. Additional information about log4j is available from the Apache Software Foundation's web site: logging.apache.org

syslog logging configuration to use org.apache.log4j.net.SyslogAppender instead.

System Logging

The Gateway's system logs record operational details about the execution of the Gateway and are intended to be used by administrators and support personnel to monitor operations and debug issues.

The system logging is controlled by the root logger configuration. The example logging configuration file that is installed with Gateway sends output to the RollingFileAppender at the INFO level. The example file is also commented with guidance on changing the configuration.

```
# Global logging configuration
log4j.rootLogger=INFO, file
```

This example shows logging directly to the Gateway server's file system, rolling log files when they reach 100MB in size, and keeping 10 generations of rolled log files.

```
log4j.rootLogger=INFO, file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.file=/var/log/caringo/cloudgateway_server.log
log4j.appender.file.maxFileSize=100MB
log4j.appender.file.maxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ISO8601} %p
[%t|%X{requestId}] %c{1}: %m%n
log4j.appender.file.threshold=DEBUG
```

This example shows sending logs to a centralized syslog server named "biglog.example.com" on the local4 facility.

```
log4j.rootLogger=INFO, syslog
log4j.appender.syslog=org.apache.log4j.net.SyslogAppender
log4j.appender.syslog.facility=local4
log4j.appender.syslog.syslogHost=biglog.example.com
log4j.appender.syslog.layout=org.apache.log4j.PatternLayout
log4j.appender.syslog.layout.ConversionPattern=%p [%t|%X{requestId}]
%c{1}: %m
log4j.appender.syslog.threshold=DEBUG
```

The log4j has many advanced configuration options and administrators should refer to the Apache Software Foundation's documentation for more details.

The rsyslog service on some operating systems does not prefix messages with time stamps in the default configuration. You may either change the rsyslog configuration to include them or you may have Gateway include a high-resolution time stamp in the message body. Adding the field %d{ISO8601} in the conversion pattern will tell Gateway to provide its own time stamp value. This is an example:

```
log4j.appender.syslog.layout.ConversionPattern=%d{ISO8601} %p
[%t|%X{requestId}] %c{1}: %m
```

Audit Logging

The audit logging data feed records client requests to the storage system in a well-defined format that is suitable for automatic processing by billing and compliance applications. The definition of this format is documented in Gateway Audit Logging. The configuration of the audit logging is described herein.

The audit logging is configured within the logging.cfg file along with the system logging configuration. The audit logging uses the audit logger name in order to separate its messages from the system messages. The name audit for the logger cannot be changed. Audit messages are logged at INFO level and each message event type can be filtered separately. Setting the log4j filter threshold to WARN, ERROR, or FATAL will disable output of that message event type.

This example shows sending the audit log messages directly to a file on the Gateway server, rolling the files when they reach 250MB, and keeping 30 generations of rolled files.

```
log4j.appender.audit=org.apache.log4j.RollingFileAppender
log4j.appender.audit.file=/var/log/caringo/cloudgateway_audit.log
log4j.appender.audit.maxFileSize=250MB
log4j.appender.audit.maxBackupIndex=30
log4j.appender.audit.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.audit.layout.ConversionPattern=%d{ISO8601}{GMT} %p
[%X{requestId}] %m%n
log4j.appender.audit.threshold=DEBUG
```

This example shows sending the logs to a centralized syslog server named "oversight.example.com" on the local3 facility.

```
# Audit file appender
log4j.appender.audit=org.apache.log4j.RollingFileAppender
log4j.appender.audit.protocol=tcp
log4j.appender.audit.host=oversight.example.com
log4j.appender.audit.port=515
log4j.appender.audit.facility=LOCAL3
log4j.appender.audit.maxMessageLength=10000
log4j.appender.audit.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.audit.layout.ConversionPattern=%d{ISO8601}{GMT} %p
[%X{requestId}] %m%n
log4j.appender.audit.threshold=TRACE
```

For direct control of the message event types, change the filtering threshold to INFO to enable and FATAL to disable the type. This example shows all message event types set to INFO in order to enable them.

```
log4j.logger.audit.scsp = INFO, audit
log4j.logger.audit.auth = INFO, audit
log4j.logger.audit.domain = INFO, audit
log4j.logger.audit.bucket = INFO, audit
log4j.logger.audit.admin = INFO, audit
```

> See Gateway Audit Logging for information about message event types and the format of the audit log messages.

Content UI Installation

- Audience
- System Requirements
- Installing the Content UI
- Required Access Policies
- Customizing the Content UI
- Upgrading the Content UI

> See also these sections:
>
> - Content UI 5 Release Notes
> - Swarm Content UI (usage guide)

Audience

This section is intended for people in the following roles.

- Storage system administrators administering a cloud of shared storage
- Cloud administrators responsible for administering domain storage allocation and policies for different user groups or applications

You need knowledge of Gateway and Storage, as well as knowledge of LDAP or Active Directory setup and configuration. Background in basic networking and knowledge of a Linux operating systems is helpful.

System Requirements

Hardware	The Content UI does not require additional hardware. The Content UI should be installed on any Gateway server that services users that will access content. If a subset of the Gateways in a load balancing group are reserved for external access (where Content UI access is not required), then you do not need to install the Content UI on that subset.
Browser	The Content UI requires a JavaScript-enabled internet browser. Development and testing were conducted using the most recent versions of Firefox and Chrome.

Swarm	Swarm 8.1 is required, which must: • Be configured to support Gateway • Have a Search feed configured and running
Gateway	One or more Gateway version 4.0 or later servers, which must: • Be correctly configured and started • Have the SCSP protocol configured for external access • Have policy.json and idsys.json files set for authentication (see Gateway Installation)

Installing the Content UI

1. Install and configure Gateway. You must ensure all storage domains have DNS entries in order to properly access them from the Content UI.

   > Important
   > Any Gateway server on which you are installing the Content UI must be configured with SCSP enabled.

2. Go to the directory where you unzipped Gateway.
3. Install the Content UI:

   ```
   yum install caringo-gateway-webui-{version}.rpm
   ```

   > Note
   > The Content UI replaces the legacy CloudScaler Admin Portal. If the old Admin Portal was previously installed on the Gateway server, it needs to be removed because it is incompatible with this release.

   To remove an old copy of the Admin Portal, run the following commands:

   ```
   rm -rf /etc/caringo/cloudscaler/portal
   rm /etc/caringo/cloudgateway/web.d/10portal.web.xml
   ```

4. Restart the Gateway.

RHEL/CentOS 6

```
service cloudgateway restart
```

RHEL/CentOS 7

```
systemctl restart cloudgateway
```

5. Navigate to the login page for the Content UI, using the base URL of any storage domain in the cluster on its configured SCSP port:

```
http://{storage-domain}/_admin/portal/
```

Required Access Policies

What is visible in the Content UI is controlled and protected by your access policy documents. For example, a policy that grants access only for a particular domain would block its members from seeing anything (domains, tenants, clusters) outside of the domain for which they are authorized. As part of implementation, policies must be set in order to grant use of the Content UI.

These are the essential permissions to allow users with no other domain-level permissions to navigate to and view a bucket and its objects:

1. GetDomain is the essential, required permission for all domain users to see their Content UI.
2. ListEtc is needed to see collections listed.
3. GetPolicy is needed to open a collection.
4. GetQuota is needed to avoid errors.

> See Setting Permissions (Access Policy editor), Gateway Access Control Policies, and the best practices in Policy Document.

Customizing the Content UI

Styling — The Content UI incorporates an empty CSS file that you can use to override the Content UI's styling, both for rebranding purposes and to protect your changes across upgrades. The customization stylesheet is `css/custom.css`.

> Tip
> The `custom.css` includes instructions for how to replace logos on the login page, headers, and About page.

Help links —You may also customize the links for Documentation and Support on the Resource Menu:

1. In a text editor, open `/opt/caringo/gateway-webui/customLinks.json`
2. Locate and add URLs for one or both of these properties:
   - `"userDocumentationLink":""`
   - `"userSupportLink":""`
3. Reload the Content UI and verify that the "Documentation" and "Online Support" links point to those that you specified in the JSON file.

Upgrading the Content UI

> See the Upgrading section in the Content Gateway Release Notes for your version.

SwarmNFS

Optional Swarm Component
This is an optional Swarm client, with its own distribution packaging and licensing.

> See also the SwarmNFS Release Notes.

- SwarmNFS Overview
- SwarmNFS Planning
- SwarmNFS Deployment
- SwarmNFS Server Installation
- SwarmNFS Export Configuration
- SwarmNFS Listings
- SwarmNFS Troubleshooting

SwarmNFS Overview

- Simplified Architecture
- Simplified Security

SwarmNFS is a lightweight protocol converter that seamlessly integrates Swarm scale-out object storage with NFS v4. It combines Swarm's universal, multi-protocol namespace with the power of enhanced and custom metadata, offering you new ways to manage, view, and analyze your data. SwarmNFS provides a traditional file interface to Swarm object storage for content generators (enterprises, researchers, web-based applications, and developers) who use native NFS-based applications to create, access, and manage that content while allowing for the same content to be created, accessed, and managed through modern cloud and object APIs such as S3 and SCSP.

SwarmNFS avoids the common problems of traditional gateway and connector file-based storage solutions: protocol and storage silos, bottlenecks, and single points of failure. SwarmNFS with Swarm object storage provides, as standard, high availability (HA), data management (from creation to expiration), powerful metadata management, and ad hoc search to your content.

Key features

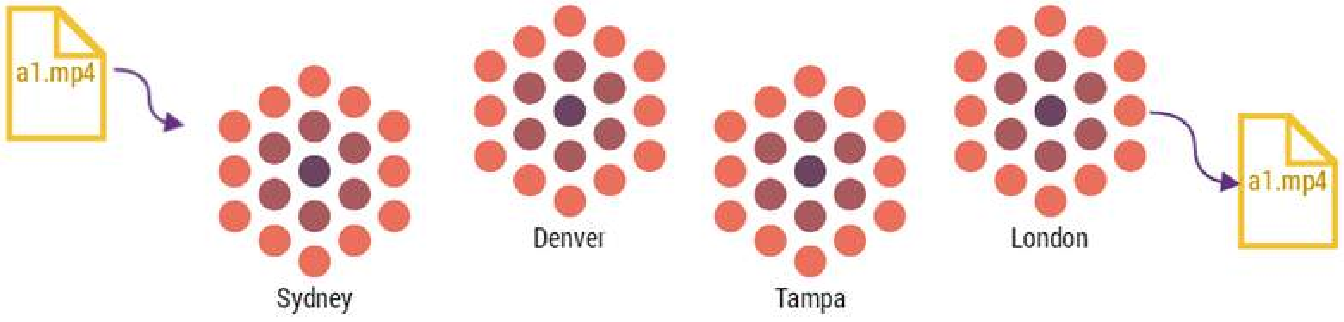
- Universal namespace — eliminates protocol silos by enabling every object to be written/accessed via NFSv4, S3, or SCSP/HTTP, all without restriction
- Lightweight for rapid deployment (VM or physical)

- Built-in active/active HA, no clustering required or limitations on number of active access points
- Direct access to read and update an object's metadata over NFS
- Stateless — reducing data loss of uncommitted resources
- Data is read from and written directly to Swarm, which means no latency and risk of data loss introduced through staging data, as with traditional gateways and connectors
- Leverages Swarm’s built-in distributed features for resilience and HA

Benefits

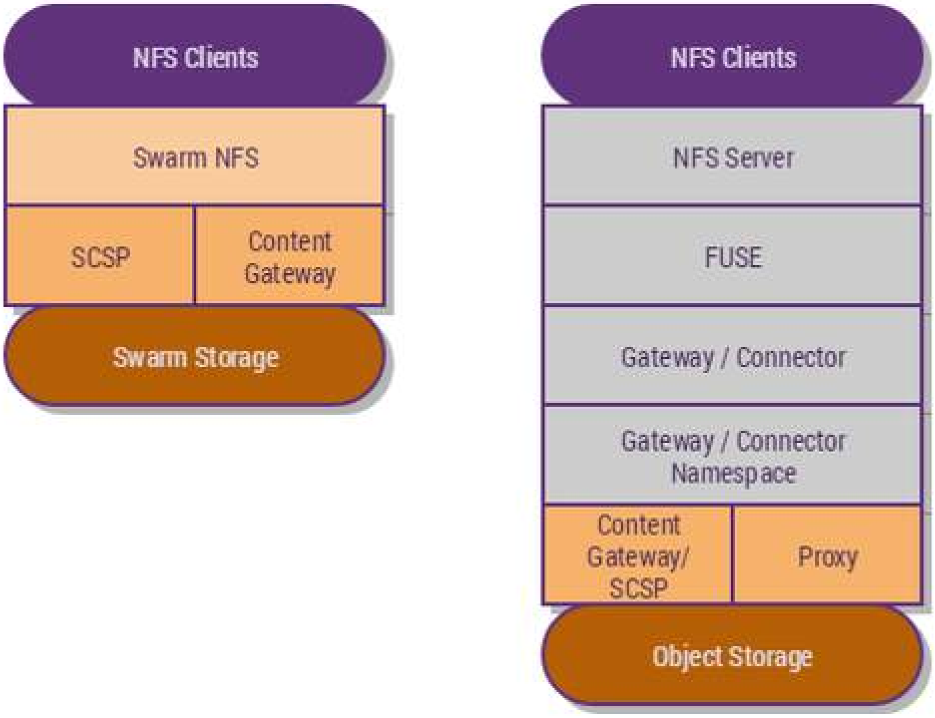
Productivity	Store, access, and manage files	• Data portability — multi-protocol in/out through NFSv4, S3, HDFS, or HTTP; read data written through FileFly • Streams data directly to and from Swarm (no local gateway storage staging or spooling) • Brings rich custom object metadata to file though NFS • Mount domains, buckets, or views that are filtered by custom object metadata
Less Risk	Security and scale with no single point of failure	• Limitless scale • Rapid scale thought physical servers, VMs or appliance • No storage or protocol silos • No read performance performance latency through data staging • Multi SwarmNFS instance managed through a single pane of glass • Security settings in Swarm propagate through all protocols • Builtin active/active HA that requires no local disk and no clustering • Auto client resume — if there is a communication issue between client and SwarmNFS, the client restarts up where it left off
Lower TCO	Leverage Swarm scale-out storage	• High availability and data protection are standard automated features • Continuous protection with seamless movement between replication and erasure coding • Eliminates the need for backups • Leverage Swarm’s automated, policy-based data management • Automatically replicate content to a remote site for distribution • Manage files from creation to expiration • WORM, Legal Hold, and Integrity seals are standard

SwarmNFS gives you true distributed file management, with the self-healing modularity of clusters that can have resources come online and go in and out of service with no disruption to user experience:



Simplified Architecture

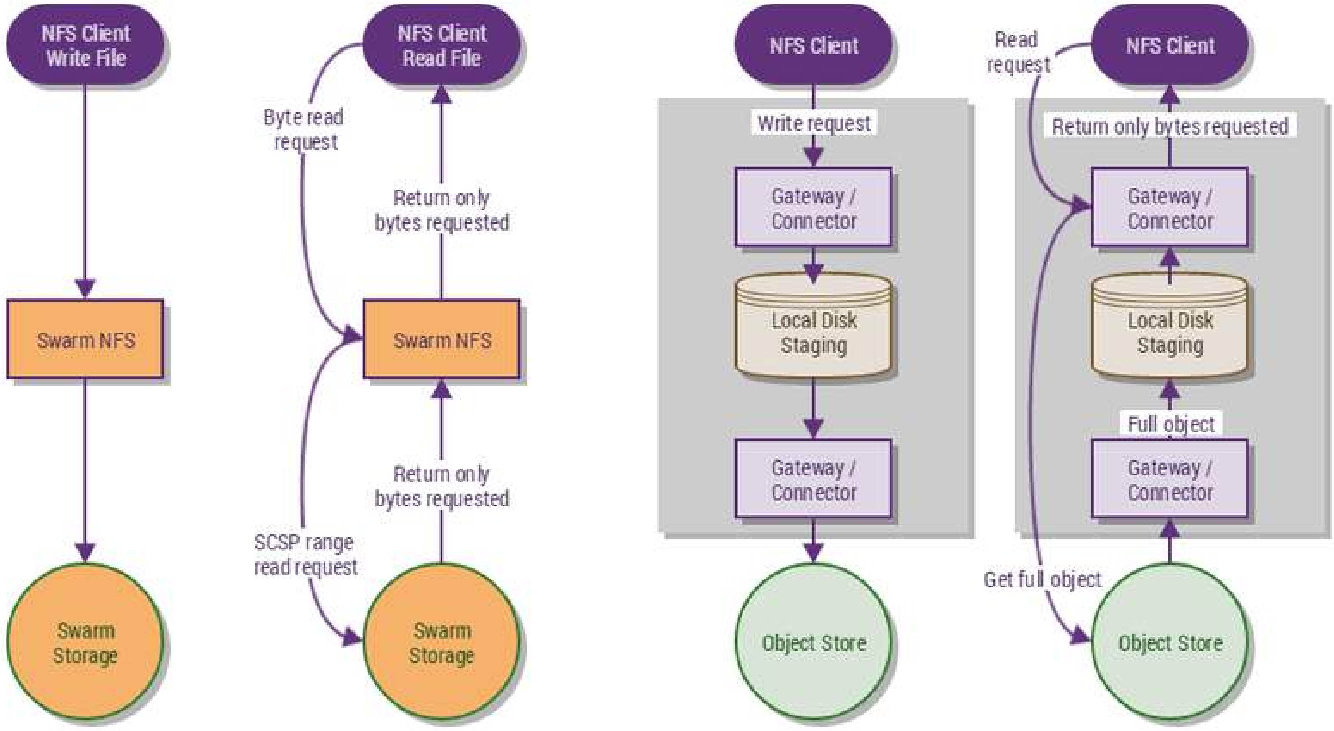
SwarmNFS is a lightweight nfs-ganesha plugin, which is simple to both install and manage. A traditional gateway or connector has many more moving parts, each adding restrictions, overhead, and complexity:



High Availability

SwarmNFS is stateless, which means that you can spin up as many SwarmNFS instances as needed and all NFS instances are active, with no failover or clustering required. With SwarmNFS, HA is simple and part of the standard product: no special configuration/clustering or management required.

Traditional gateways and connectors must stage objects as files locally on the gateway, which means that you are limited by disk space on gateway server. SwarmNFS does not cache or stage files/objects on local disk space; rather, it streams data directly to and from Swarm. You escape the performance overhead of writing complete objects to a local gateway staging disk, and you have no risk of losing data if the Gateway crashes before data is spooled off the gateway to the object store.



Note
Given the stateless nature of Swarm and SwarmNFS, file locking exists only within a single SwarmNFS server.

Simplified Security

POSIX — SwarmNFS adds basic POSIX security to the modern object security inherent in Swarm. This is important as it means that you achieve uniform object security through and across NFS, SCSP, and S3, with additional POSIX file security when objects are accessed though SwarmNFS.

Access Control — SwarmNFS supports basic POSIX UNIX-style ACLs (user, group, other); object access control is managed via Content Gateway. SwarmNFS validates the login/password, and it then leaves Gateway to control object access. Native Swarm access means Anonymous NFS access to objects only.

SwarmNFS Planning

- Planning the SwarmNFS Environment
- Planning the SwarmNFS Configuration

SwarmNFS can co-exist with other applications running on the same Linux server, although SwarmNFS expects to have sole ownership of its assigned ports and resources. SwarmNFS can be deployed onto the same Linux server as the Content Gateway, or it can run on its own dedicated operating system instance.

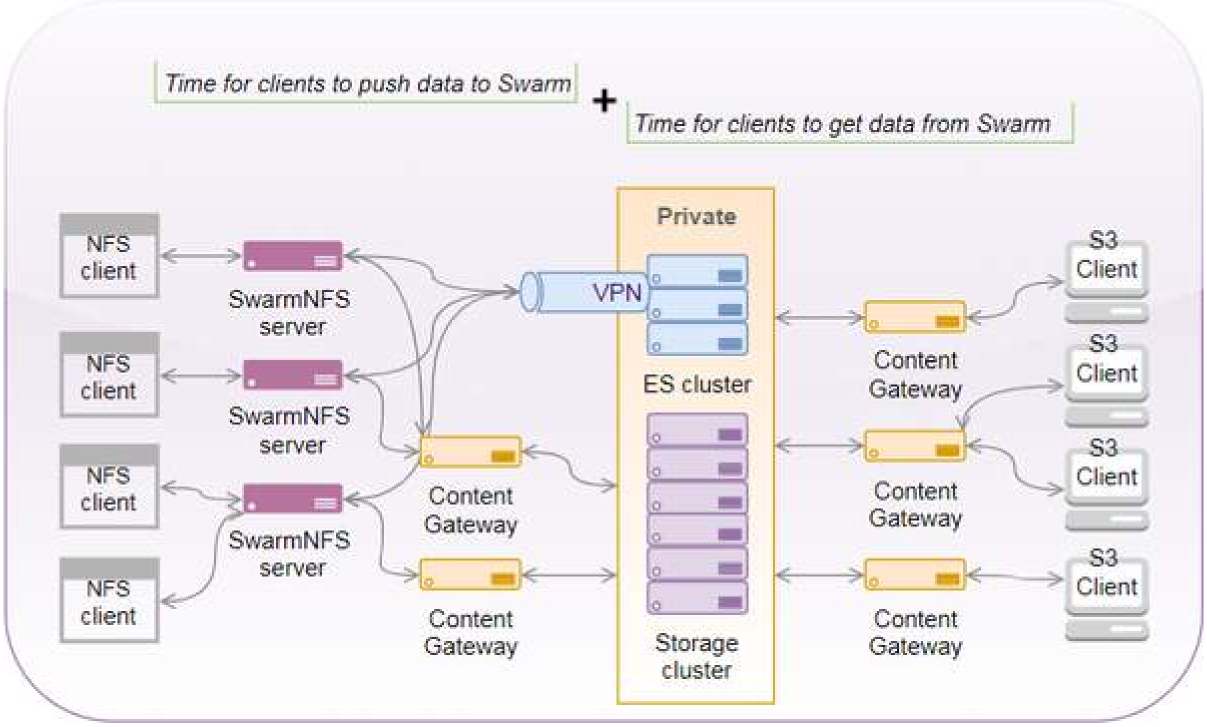
- Stateless — SwarmNFS is stateless, so each process stores no run-time configuration outside of volatile RAM locally, with the exception of a basic configuration file read on startup (see SwarmNFS Export Configuration).

Should a process fail or restart, any incomplete operations that it was processing are lost, and it will restart clean and stateless.

- Multiple active — There is no limit to the number of SwarmNFS servers that can be online at any time: multiple SwarmNFS processes can be active running on different servers simultaneously. SwarmNFS running on different servers can be configured identically and present the same object view as other active running SwarmNFS. SwarmNFS servers can be added and removed independently of the others, and any one SwarmNFS server going offline will have no effect on other running instances.

The job of an NFS server is to keep data and metadata well stored and to move it efficiently; the job of an NFS client is to translate and adapt the NFS protocol to its local environment efficiently, which is harder. Client performance is complex, and it depends completely on the performance of servers and their drives, as well as the efficiency of networking, caching, and data structures.

Keep in mind that many areas affect the throughput that you can achieve, from writing data via an NFS client through its ingest in Swarm Storage to its ability to be read by a Swarm client such as S3.



Planning the SwarmNFS Environment

Component	Value	Notes
Is the Swarm cluster already running?	yes | no	Swarm Must be running before SwarmNFS can be configured
Is a Elasticsearch cluster installed and running?	yes | no	Elasticsearch must be running before SwarmNFS can be configured

Is there a working Swarm “Search Full Metadata” feed to the Elasticsearch cluster that SwarmNFS will use?	yes | no	SwarmNFS requires working a full metadata search feed
Will Content Gateway be used? if so, is it already running?	yes | no	If (optional) Content Gateway is to be used then it must be running before SwarmNFS can be configured
Is port 91 (default) open between SwarmNFS and the storage nodes or Gateway?	yes | no	SwarmNFS must be able to connect to the Swarm management API, either directly to swarm or via the gateway proxy
Is port 9200 (default) open between SwarmNFS and the Elasticsearch nodes?	yes | no	SwarmNFS needs to be able to connect directly to port 9200 on the Elasticsearch Nodes
How many SwarmNFS servers will be installed?		You can have one or more SwarmNFS Servers
Will all SwarmNFS servers present the same NFS exports?	yes | no	SwarmNFS servers can present the same exports, or different exports, you can also have groups of SwarmNFS servers presenting different exports
Which operating system will SwarmNFS be installed on?	CentOS | RHEL	If using RHEL (Red Hat Enterprise Linux), your site needs a Red Hat license
If SELinux is enabled and enforcing on the SwarmNFS servers, has SELinux been configured to allow SwarmNFS to initiate connections to Elasticsearch and the Swarm Management API?	yes | no	If SELinux is enabled, it must allow SwarmNFS to open network connections to Elasticsearch (default port 9200) and the Swarm Management API (default port 91)

Planning the SwarmNFS Configuration

Plan for how SwarmNFS will be configured:

Component	Examples	Notes

Swarm Nodes	`192.168.1.30, 192.168.1.38, 192.168.1.44`
	`swarmnode1.example.com, swarmnode2.example.com, swarmnode3.example.com`
		At least three IP addresses or DNS-resolvable names
		Not needed if connecting via Content Gateway

Content Gateways	`172.30.26.2, 173.20.26.3`
	`contentgateway1.example.com, contentgateway2.example.com`
		One or more IP addresses or DNS-resolvable names
		Not needed if connecting direct to Swarm

Elasticsearch Nodes	`172.30.26.20, 173.20.26.21`
	`es1.example.com, es2.example.com`
		One or more IP addresses or DNS-resolvable names

SwarmNFS Deployment

- Swarm Software Requirements
- Implementing SwarmNFS
- Optimizing Performance
- Mounting the Exports

Swarm Software Requirements

These are the Swarm packages and minimum versions that work with and comprise SwarmNFS. The components are bundled and available on the Swarm download page on connect.caringo.com.

Component	Version	Package Name	Requirements
Storage	9.5.0+	caringo-storage-{version}-{rel}.noarch.rpm	Enable Erasure Coding (EC) — see g Content Policies Enable Overlay Index: `index.ove` `ed= true` — see Configuring the Index Enable Replicate on Write (ROW) iguring ROW Replicate On Write Set `ec.segmentConsolidationF` `=100`
Storage UI	1.2.3+	caringo-storage-webui-{version}-{rel}.noarch.rpm	Set the Swarm Search feed to hav second Batch Timeout (see Viewi Editing Feeds)
Elasticsearch	2.3.3+	elasticsearch-{version}.rpm elasticsearch-curator-{version}.rpm	In the Elasticsearch configuration `lasticsearch.yml)`, make these • comment out: `filter: lowe` e-insensitive metadata search Swarm is incompatible with S • add: `script.inline: true` • add: `script.indexed: true`
Search	2.5.2+	caringo-elasticsearch-search-{version}-{rel}.noarch.rpm	SwarmNFS 2.0 requires the latest RPM. (SNFS-437)
Metrics	2.5.1+	caringo-elasticsearch-metrics-{version}-{rel}.noarch.rpm	(optional)
Gateway	5.2.3+	caringo-gateway-{version}-{rel}.x86_64.rpm	SwarmNFS can be deployed onto Linux server as the Content Gatev

Content UI	5.4.0+	caringo-gateway-webui-{version}-{rel}.noarch.rpm	(optional)
NFS	2.0+	caringo-nfs-{version}-{rel}.x86_64.rpm caringo-nfs-libs-{version}-{rel}.x86_64.rpm	Important Do not install SwarmNFS se the same host as Elasticse

Implementing SwarmNFS

> Important
>
> Before proceeding, complete your SwarmNFS Planning.

For SwarmNFS, do the following:

- Install one or more SwarmNFS servers for NFS 4 on designated hardware. (See SwarmNFS Server Installation.)
- Create the exports needed for your implementation (See SwarmNFS Export Configuration.)
- For functional verification and troubleshooting, create a test domain and bucket and then create an export for that bucket. (See Configuring Domains, Configuring Buckets.)
- For each of your SwarmNFS exports, conduct basic testing of read, write, and delete using your NFS client mounts.

Optimizing Performance

For best performance, the export's Read Buffer Size must closely approximate the workload that you expect on that share (see SwarmNFS Export Configuration). The default is 12MB for general workloads.

- Lower read-ahead buffer size if most reads will be small and non-sequential.
- Increase read-ahead buffer size if most reads will be large and sequential (read-ahead buffer sizes of up to 32MB have been tested).

  > Tip
  >
  > You can export the same bucket more than once, each with a different read-buffer size. You can then point your clients and applications to the share that best matches their workload.

Mounting the Exports

When mounting your SwarmNFS exports, follow these guidelines:

Linux	Mount the exports as normal, with these explicit options:
	• Increase the timeout, timeo, to 9000.
	• Enable noatime (no access time), which lets the system skip updating the file system for files that are simply being read. This has no effect on the write time information, which is always updated with every write to the file.
	• To ensure that you mount using the expected protocol, add the "`-t nfs`" and "`vers=<nfsvers>`" mount options.
	Best practice: Mount using NFS v4.1. If your client does not support 4.1, then fall back to 4.0.
	NFS v4.1
	`mount -t nfs -o timeo=9000,noatime,vers=4.1 SwarmNFSserver:/ /mnt/SwarmNFS`
	NFS v4.0
	`mount -t nfs -o timeo=9000,noatime,vers=4 SwarmNFSserver:/ /mnt/SwarmNFS`

OS X	Similar to Linux, but the client is restricted to NFS v4.0 and there are version-specific issues. See the article Mounting exports on macOS.
Windows	You cannot mount SwarmNFS to Windows because it has no NFS 4.x client.

SwarmNFS Server Installation

- System Requirements
- Installing SwarmNFS

System Requirements

These are minimum requirements for SwarmNFS servers in production:

OS	RHEL/CentOS 7	CentOS 7.4+ SELinux Policy Management:
		Edit `/etc/selinux/config` and set `SELINUX=permissive`, or map the ports needed by Swarm using `semanage-port`:
		`semanage port -a -t http_port_t -p tcp 91`
		`semanage port -a -t http_port_t -p tcp 9200`
		`semanage port -a -t http_port_t -p tcp 9300`

CPU	2 cores	
RAM	4 GB (minimum)	4GB for a single export Add an extra 4GB for each additional export when using default export memory settings
Drive	Dedicated /var/log partition	Provide at least 30 GB for SwarmNFS logs

The RAM requirements increase with the number of exports, but the greatest impact on CPU and RAM is driven by the number of concurrent client operations. In general, the more concurrent client operations that are being served, the more RAM and CPU cores you need. How much and how many depends on what the clients are handling (in terms of file/object sizes), so focus on whether the allocated resources are being fully utilized. As they near full utilization, add more.

> Important
> - Where write performance is critical, install the SwarmNFS server on physical hardware (not as a VM).
> - Paging hurts performance. If paging out does occur, increase RAM rather than increasing disk space for paging.
> - When mounting exports directly on the SwarmNFS server, do not enable verbose (DEBUG) logging.

Installing SwarmNFS

The RPM for SwarmNFS includes an interactive script for completing the needed SwarmNFS configuration for a specific Swarm Storage cluster. The script prompts you for the URL of the SwarmNFS JSON export configuration created for Swarm Storage, so the NFS exports must be defined via the Swarm UI.

> Note
> The script enables core file generation; it can be disabled via the `nfs-ganesha.service` file (`/usr/lib/systemd/system/`) or through the system-wide configuration.

Before installing any SwarmNFS servers, prepare the export configuration files that they will need to reference:

1. In the Swarm UI, select Settings > NFS.

2. Define one or more server groups (each group having one export configuration URL to be shared among a set of servers).
   See SwarmNFS Export Configuration.
3. Within each group, define one or more exports, which will become the mount points for your applications.

   > Tip
   > The export URL is non-functional but valid before you define exports, and you can add to and update your exports at any point.

On each CentOS 7 system that you want to be a SwarmNFS server, run the scripted process:

1. Download the SwarmNFS package from Caringo Connect.
2. Install the EPEL release, which has the needed packages for NFS:

   ```
   yum -y install epel-release
   ```

3. Install the Caringo RPMs:

   ```
   yum install caringo-nfs-libs-<version>.rpm
   yum install caringo-nfs-<version>.rpm
   ```

4. Run the SwarmNFS configuration shell script (which is in the path and located in `/usr/bin`) as follows:
   a. Copy in the export configuration URL from the UI: `http://172.30.13.95:91/api/nfs/configurations/_plain1`
   b. Add the user and password to be used for authentication:

      ```
      SwarmNFS-config nfsadmin@password
      http://172.30.13.95:91/api/nfs/configurations/_plain1
      ```

      The script generates the local SwarmNFS service configuration, validates the environment, enables the SwarmNFS services, and then starts the SwarmNFS services. If the NFS service fails, it is configured to restart.
5. To allow SwarmNFS to start automatically on boot, enable the service:

   ```
   systemctl enable /usr/lib/systemd/system/nfs-ganesha.service
   ```

6. To verify the status of the services, run this command:

```
systemctl status nfs-ganesha
```

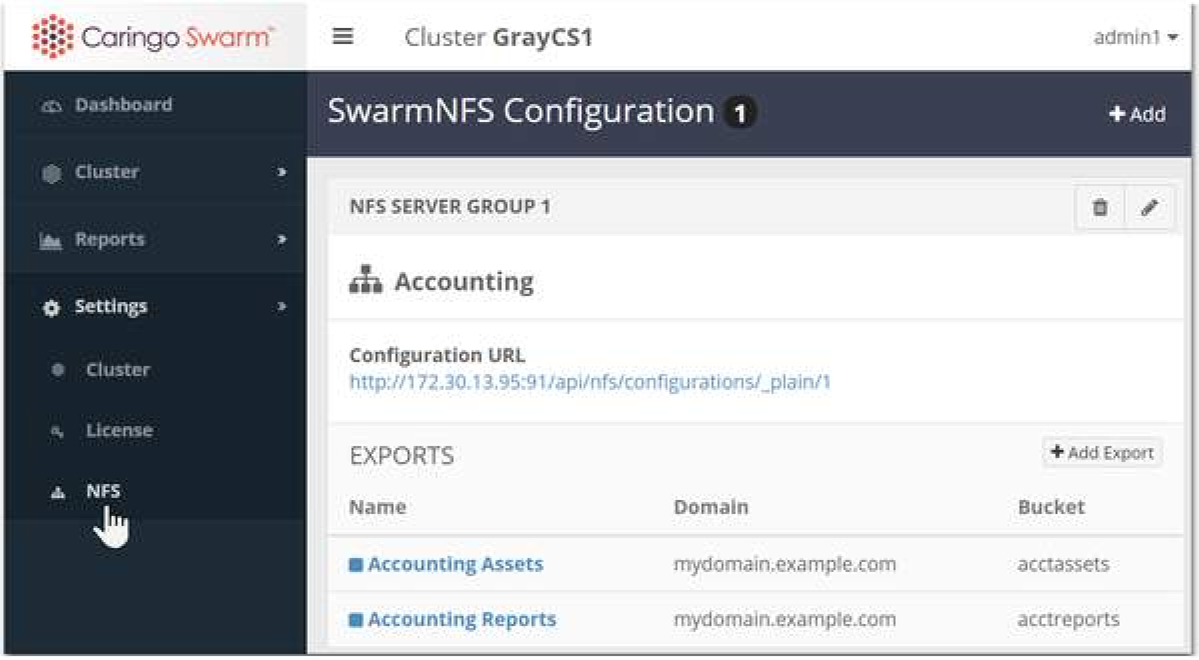
This status report is comprehensive and includes which processes are running.

> Tip
>
> On startup, SwarmNFS may generate WARN level messages about configuration file parameters. These are harmless and can be ignored. (SNFS-216)

SwarmNFS Export Configuration

The NFS page in the Swarm Storage UI lets you create and manage your NFS server groups and exports.

> Important
>
> You must create the storage cluster's default domain before configuring SwarmNFS. This domain has the same name as the `cluster.name` setting's value. The domain can be created with the Content UI or an HTTP utility like curl.



SwarmNFS Configuration 1

NFS SERVER GROUP 1

Accounting

Configuration URL
http://172.30.13.95:91/api/nfs/configurations/_plain/1

EXPORTS

Name	Domain	Bucket
Accounting Assets	mydomain.example.com	acctassets
Accounting Reports	mydomain.example.com	acctreports

You can create separate groups (sets) of SwarmNFS that are configured in pools; this lets you support different clients and optimize for different roles. You can also set some configuration settings locally, to override global configuration settings.

Why different server groups? These are typical configuration changes that you might want to make for specific servers:

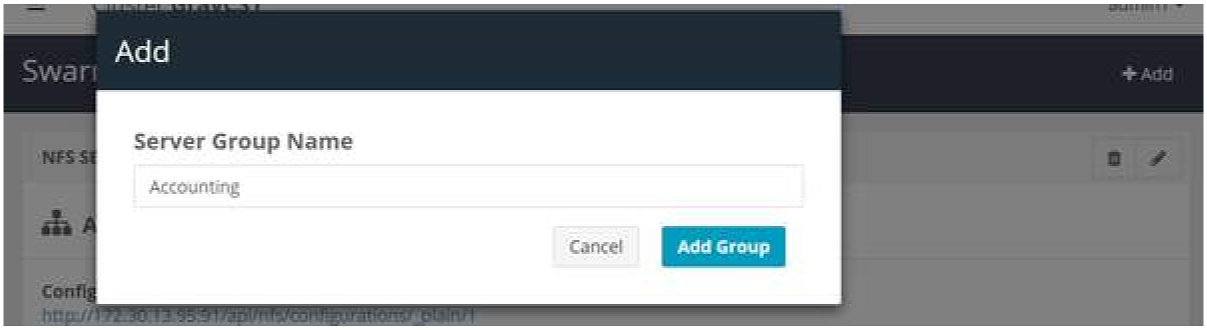
- Include DEBUG level logging
- Change the log file location
- Add local resource restrictions
- Change interface or IP address bindings
- Reduce maximum threads or open/concurrent client connections

While every SwarmNFS server will retrieve the global configuration file stored within Swarm, each server group can optionally override the global settings with their own configuration file.

> Important
> Restart NFS services after making any configuration changes. The NFS server does not yet support dynamic updates to the running configuration.

Adding Server Groups

Server Groups are created with the + Add button at top right.

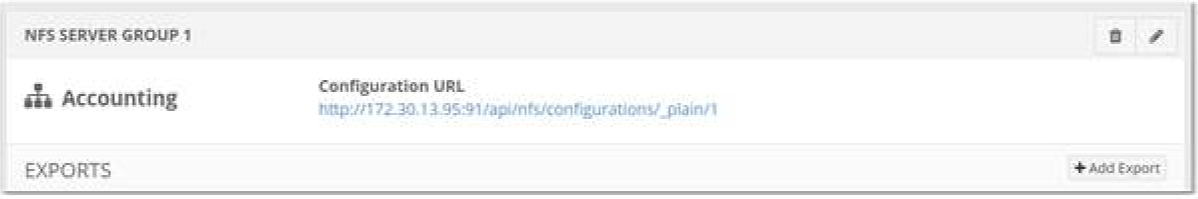


Add

Server Group Name: Accounting

> Best practice
>
> Before creating a Server Group, be sure that your default domain is specified and to verify the existence of the domain and bucket that will be defined in the scope.

The resulting group is a container for exports that will share a common configuration:



NFS SERVER GROUP 1

Accounting

Configuration URL: http://172.30.13.95:91/api/nfs/configurations/_plain/1

EXPORTS

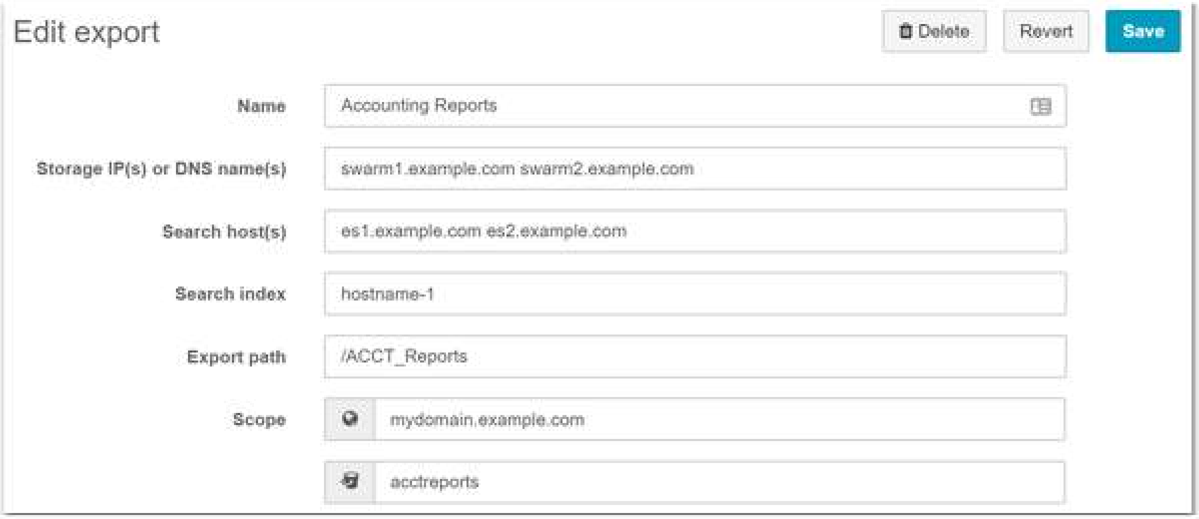
Name	When you add a Server Group, you only supply a name, which is a description; the unique identifier is the count at the end of the Configuration URL. The new group appears at or near the end of the listing, ready to be configured with exports.
Configuration URL	Each NFS Server Group has a unique Configuration URL, which you can click to view the current export definitions. These are the auto-generated and auto-maintained JSON settings being stored by Swarm for the group. Note The configuration is empty until you add one or more exports.

> Important
>
> Although group configurations may be shared across NFS servers, each server must be configured with only one group.

Adding Exports

Each SwarmNFS export that you create with +Add Export goes direct to Swarm and uses anonymous/guest authentication.

> Note
> Each export is specific to one and only one Swarm bucket, but clients viewing the mounted directory will be able to view, create, and use virtual directories within it via the prefix feature of Swarm named objects (`myvirtualdirectory/myobjectname.jpg`).



Edit export

Name: Accounting Reports
Storage IP(s) or DNS name(s): swarm1.example.com swarm2.example.com
Search host(s): es1.example.com es2.example.com
Search index: hostname-1
Export path: /ACCT_Reports
Scope: mydomain.example.com
acctreports

Name		Unique name for your export, to distinguish it from the others in Swarm UI listings.
Storage IP(s) or DNS name(s)		The IP address(es) or DNS-resolvable hostname(s) for one or more Swarm Storage nodes.
Search host(s)		The IP address(es) or DNS-resolvable hostname(s) for one or more Swarm Elasticsearch servers. Note: Both Gateway and SwarmNFS use the Primary (default) search feed. If a new feed is made Primary, these servers must be restarted.
Search index		Optional, but recommended for larger sites to improve performance. The unique alias name of the Primary (default) search feed. Locate this value as the Alias field in the primary search feed's definition. (v1.2.4)
Export path		Case-sensitive. Unique pseudo filesystem path for the NFS export. Cannot be set to a single slash ("`/`"), which is reserved.
Scope	Domain Bucket	Specifies where the data written via the export will be associated: which domain and bucket to use. Important: Be sure to verify the existence of the domain and bucket that you specify here.

Storage interface	Gateway	When you deploy in a Gateway environment, you can use pass-through authentication, which means authenticating to Gateway using the same login and password that was provided for authentication by the client to SwarmNFS. You also have the choice of session tokens (with various expirations) and single user authentication, by login credentials or token. SwarmNFS maintains exactly the same level of object security when accessing or modifying objects through SwarmNFS or other protocol such as SCSP, S3 or the SwarmFS (Hadoop). Gateway provides security at domain and bucket level only and objects inherit those security policies, accessibility to all unnamed objects are restricted to that of the user's rights at the containing domain, and restricted to rights set at the containing bucket level for named objects. SwarmNFS layers no individual object security (named or unnamed) above that enforceable by Gateway.
	Direct to Swarm	When you deploy without Gateway, SwarmNFS uses anonymous/guest authentication.

Cloud Security (Gateway only)

If you select Content Gateway to be your Storage interface, you see an additional section called Cloud Security.

> Tip
> Each SwarmNFS export that you create to use the Content Gateway can have an entirely different security method, as needed by its use case.

Session Token	Token Admin Credentials by Login Token Admin Credentials by Token	User, Password, Expiration Token, Expiration
Single User	Authenticate by Login Authenticate by Token	User, Password Token
Pass-through / None	n/a	

Client Access

This optional section allows you to customize access control both globally (for this export) and for specific clients.

Access type	Defaults to full read/write access. These other access restrictions are available: • All operations (RW) - default • No access (None) • Read-only (RO) • No read/write (MDONLY) - allows listing and metadata updates without access to file contents • No read/write/modify (MDONLY_RO) - allows listing but no metadata updates and no access to file contents

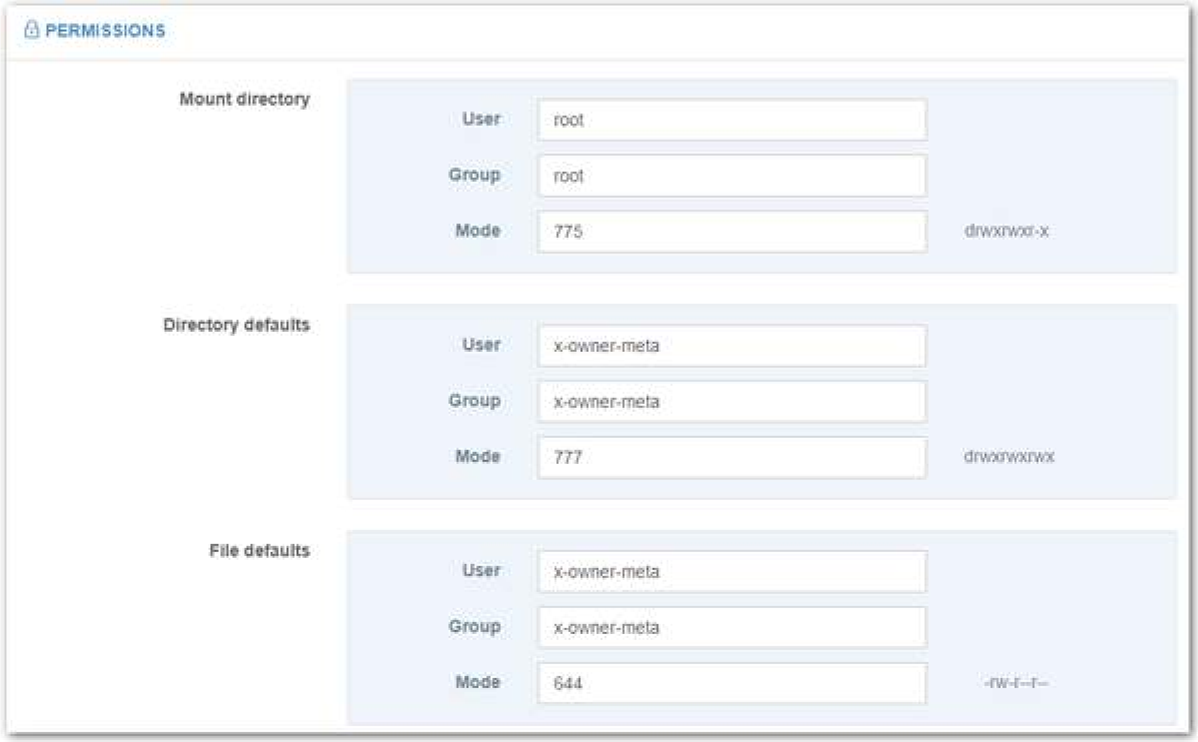
Squash	Defaults to no squashing (allows all user IDs). • None - default • Root - squashes the remote superuser (root, uid=0) when using identity authentication (local user is the same as remote user) • All - squashes every remote user, including root.
Squash user id (uid) mapping Squash id (uid) mapping	User ID and Group ID can be set when you have the NFS server authenticating users from a different authentication sources and/or you want all the files to have a consistent user/group. Typical situations: • All clients are configured to use local password/group files, but SwarmNFS through the Content Gateway is configured to use LDAP. • All clients have local password/group files, but some users may not exist on all clients systems or may differ on each client. • All clients have the same users and groups, but they were created in a different order. • All clients authenticate using individual logins/accounts, but you want all files to have the same consistent owner and group regardless of the user reading or writing the files. • A client mounts the NFS exports as anonymous, but you want the files presented over the share to all NFS clients to have a consistent UID and GID.
Client(s)	As needed, customize the access for one or more specific clients. Note: These override the settings specified above, if any.

Permissions

Files and directories in a SwarmNFS system support standard Unix-style read/write/execute permissions based on the user and group IDs asserted by the mounting NFS client. When your users try to access files and directories, SwarmNFS checks their IDs to verify that they have permission to access the objects, and it uses these IDs as the owner and group owner for any new files and directories that they create.

For each export, you can customize the default User, Group and ACL Mode for the export mount, directories, and files. These settings only apply for externally created objects and synthetic folders that do not already have POSIX permissions attached to the object as standardized metadata. User and Group are ASCII values, not numeric IDs.

> Tip
>
> The ACL mode must be entered as an octal, such as 664 or 0664. Use http://permissions-calculator.org/ to generate the octal code that corresponds to the read/write/execute permissions that you want to apply.



PERMISSIONS

Mount directory

User root

Group root

Mode 775 drwxrwxr-x

Directory defaults

User x-owner-meta

Group x-owner-meta

Mode 777 drwxrwxrwx

File defaults

User x-owner-meta

Group x-owner-meta

Mode 644 -rw-r--r--

> Using x-owner-meta
>
> What you select for the export's interface and access method determines whether you should use `x-owner-meta`. Using defaults of `x-owner-meta` and 0755 or 0644 are valid only when Storage Interface is set to "Content Gateway" and the Cloud Security Access method is set to "Session Token". For all the other methods (such as "Direct to Swarm", "Single User", "Pass-through / None"), the NFS client cannot map `x-owner-meta` to a local UNIX/POSIX user.

Advanced Settings

> Important
>
> Use these recommended defaults for all of the Advanced Settings unless otherwise advised by Caringo Support.

Transport protocol	TCP	Supported transport protocol (TCP/UDP | TCP | UDP)
Storage port	80	Required. Network port for traffic to Swarm Storage nodes
Search port	9200	Required. Network port for traffic to Swarm Search nodes
Security	sys	Remote Procedure Call (RPC) security type (sys | krb5 | krb5i | krb5p)

Maximum storage connections	100	Maximum number of open connections to Swarm Storage. (v2.0)
Retries	5	(positive integer) How many times SwarmNFS will retry unsuccessful requests to Swarm and Swarm Search before giving up.
Retries timeout	90	(seconds) How long SwarmNFS will wait before timing out Swarm retries.
Request timeout	90	(seconds) How long SwarmNFS will wait before timing out Swarm requests. For best results, set this timeout to at least twice the value of the Storage setting scsp .keepAliveInterval.
Pool timeout	300	(seconds) How long discovered Swarm storage nodes are remembered.
Write timeout	90	(seconds) How long SwarmNFS will wait for a write to Swarm to complete before retrying.
Read buffer size	16000000	(bytes) The amount of data to be read each time from Swarm. If the read size buffer is larger than the client request size, then the difference will be cached by SwarmNFS, and the next client read request will be served directly from cache, if possible. Set to 0 to disable read ahead buffering
Maximum buffer memory	2000000000	(bytes) Defaults to 2GB. Maximum limit that can be allocated for the export's export buffer pool. Once exceeded, client requests will temporary be blocked until total buffers falls back below this number. (v2.0)
Buffer high watermark	1500000000	(bytes) Once the allocated export buffers reach this watermark, SwarmNFS will start to free buffers in an attempt to stay below “Maximum Memory Buffers”. During this time, client requests may be delayed. (v2.0)
File access time policy	"relatime"	Policy for when to update a file's access time stamp (atime). (v2.0) • “noatime”: Disables atime updates. • “relatime”: Updates atime only if it is earlier than last modified time, so that it updates only once after each write. • “strictatime”: Updates atime on every read and close.

SwarmNFS Listings

Each SwarmNFS export you specify presents the named objects that are stored within a Swarm bucket. For manageability, the exact objects that a client sees when connected to SwarmNFS is filtered, and virtual hierarchies are presented as much as possible.

Note

Listing delay — Objects created in Swarm natively (not via SwarmNFS) can take up to 5 minutes plus the Search Feed's Batch Timeout to appear in SwarmNFS listings, because they must be indexed by Elasticsearch. For best listing performance, lower the search feed's Batch Timeout to 1 or 0 (recommended).

Listing limit — How many entries are returned in a folder listing is limited to the value configured in Elasticsearch for "index.max_result_window". SwarmNFS will allow new files to be created even if the number would exceed "index.max_result_window", but, over time, the number of entries listed will fall back down to that configured maximum. See Configuring Elasticsearch.

Exclusive opens — SwarmNFS supports exclusive opens of a file (O_EXCL and O_CREATE) but does not support exclusive reopens (EXCLUSIVE4).

Named Object Listings

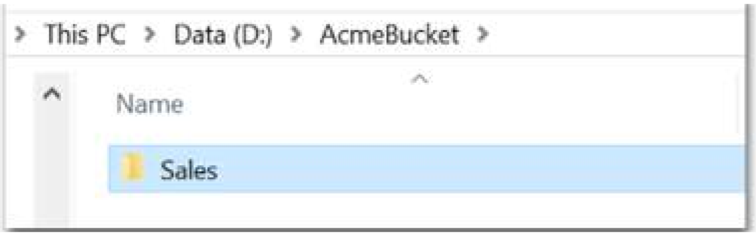
To present named object listings as if they were in a traditional file system, SwarmNFS creates a simulation: it translates each forward slash (/) in any object name into a traditional directory delimiter. SwarmNFS then presents a view of objects at the simulated directory level. For example, suppose these named objects exist in the bucket “`AcmeBucket`”:

```
/Sales/Leads/campaigns/2016/Jan/list.xls
/Sales/Leads/campaigns/2016/Feb/list.xls
/Sales/Leads/campaigns/addword.xls
/Sales/Leads/campaigns/partners.xls
```

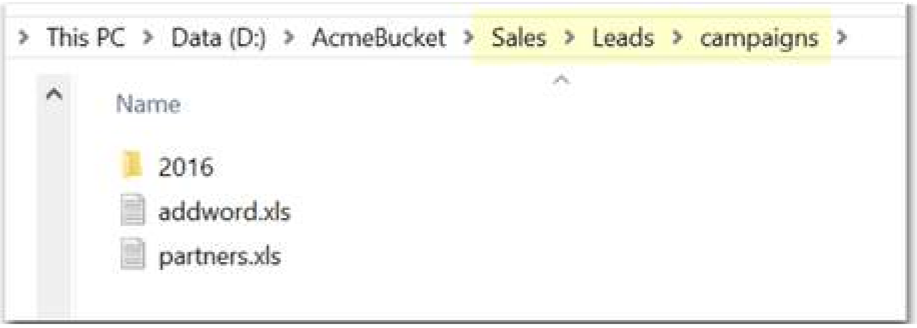
Note

Objects in Swarm Storage use forward slashes in their names for directory delimiting; SwarmNFS converts these to backslashes inline during listing transfers only if a client expects backslashes to be the directory delimiter. SwarmNFS always converts backslashes to forward slashes when communicating with Swarm Storage (including through Gateway).

By default, for a client request for a bucket-level listing, SwarmNFS provides a simulated view of the first level of contents within the bucket:

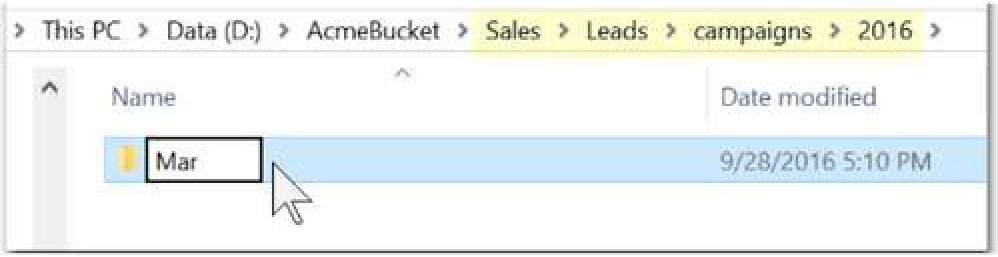


Each client request to open a specific directory level within the bucket (such as `/Sales/Leads/campaigns`) returns a new listing for that context:


> This PC > Data (D:) > AcmeBucket > Sales > Leads > campaigns >
Name
2016
addword.xls
partners.xls

If the user requests a listing for a file (such as `Sales/Leads/campaigns/addword.xls`) then only that single file would be returned.

If an end user creates a new directory (such as `AcmeBucket/Sales/Leads/campaigns/2016/Mar`), SwarmNFS creates a new empty object with Directory attribute metadata, which marks it as a simulated directory.


> This PC > Data (D:) > AcmeBucket > Sales > Leads > campaigns > 2016 >
Name
Date modified
Mar
9/28/2016 5:10 PM

> Important
>
> To delete a directory, end users must first remove any files within it.
>
> To rename a directory, end users must create a directory with the new name, move the files into it, and then delete the directory with the old name. (NFS-607)

Keep in mind that simulated directory objects only exist to support the expectations of end users and applications: objects can be written to a new virtual directory just by including it in the pathname.

SwarmNFS Troubleshooting

- Changing Logging Levels
- Failure to Load Export Configuration
- Client Mounts with Empty Listings
- Listing Exports and Clients
- Matching Requests between SwarmNFS and Storage Logs
- Missing Custom Header
- Users Lost Permissions

> Required
>
> To use ganesha_mgr for these troubleshooting steps, first install the RPM package nfs-ganesha-utils.

Changing Logging Levels

By default, SwarmNFS logs to `/var/log/ganesha.log`. To optimize read performance, the logging level for SwarmNFS defaults to `NIV_EVENT`.

Find level — To determine the current log level for the SwarmNFS plugin or all ganesha components, run the appropriate command:

```
ganesha_mgr get_log COMPONENT_FSAL
ganesha_mgr get_log COMPONENT_ALL
```

Change level — To change the logging level permanently, edit the `/etc/sysconfig/ganesha` file. These are supported levels:

- `NIV_EVENT` — SwarmNFS default, for best performance.
- `NIV_INFO` — Prints all logs below its level, such as `NIV_FATAL, NIV_MAJ, NIV_CRIT, NIV_WARN,` and `NIV_EVENT`.
- `FULL_DEBUG` — Enable for troubleshooting only.
  Best practice: Enable debug temporarily without restarting ganesha using these commands:
  - Start debug — To enable debug logging for the SwarmNFS plugin or all ganesha components, run the appropriate command:

    ```
    ganesha_mgr set_log COMPONENT_FSAL FULL_DEBUG
    ganesha_mgr set_log COMPONENT_ALL FULL_DEBUG
    ```

    > Note
    >
    > `COMPONENT_ALL` is the default for components that have not had their individual log level set.

  - Stop debug — To turn off debug logging for the SwarmNFS plugin or all ganesha components, run the appropriate command:

    ```
    ganesha_mgr set_log COMPONENT_FSAL NIV_EVENT
    ganesha_mgr set_log COMPONENT_ALL NIV_EVENT
    ```

Failure to Load Export Configuration

If, after starting ganesha, client root export mounts `[mount {server}:/ {/mntpoint}]` only list `/bkt`, SwarmNFS may not be loading its configuration.

1. Start ganesha manually in the foreground.

```
ganesha.nfsd -F
```

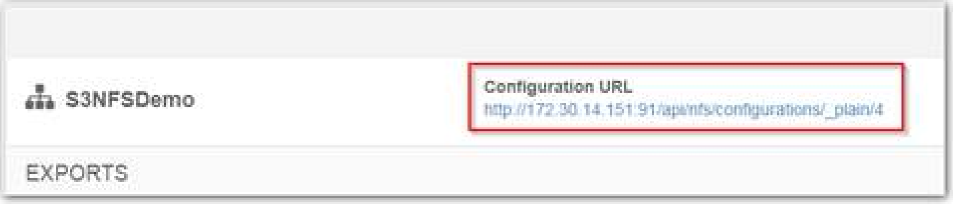
2. Wait 20 seconds. If all is working, expect to see something similar to the following:

   ```
   16:42:24,622231160 P   8186-0x7f5843c32100  libswarmio   | ###
   Registering swarmio logsink ###
   Remove Export with id 1
   Remove Export with id 2
   Add Export in /etc/ganesha/318a2790-5a1b-11e8-a1ac-002590eb7394.tmp
   Returns: status = True, Done: 1 exports added
   Remove Export with id 1
   Remove Export with id 3
   Add Export in /etc/ganesha/318a459a-5a1b-11e8-a1
   ```

3. Look for one set of Remove Export with id 1, Remove Export with x, and Add Export for each of your configured exports. If you do not see these complete sets, proceed.
4. Verify that SwarmNFS can retrieve its central configuration:

   ```
   grep Configuration /etc/ganesha/ganesha.conf
   ```

5. In the Swarm UI, go to Settings > NFS and locate the Configuration URL:

   S3NFSDemo

   Configuration URL
   http://172.30.14.151:91/api/nfs/configurations/_plain/4

6. Use curl to verify that you can manually retrieve the configuration file:

   ```
   curl -L -v {URL}
   curl -L -v http://172.30.14.151:91/api/nfs/configurations/_plain/4
   ```

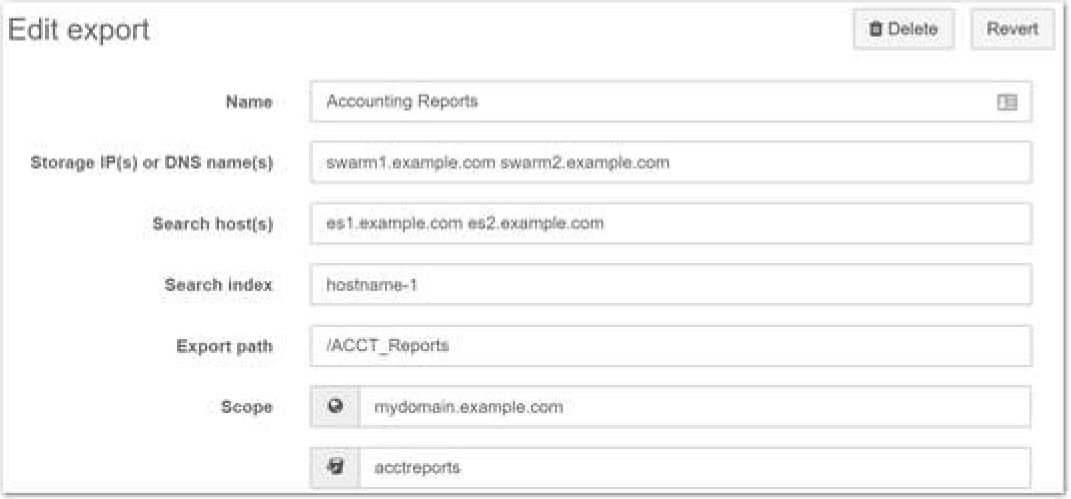
7. If you cannot manually retrieve the configuration file via curl, resolve the issue and then restart ganesha manually in the foreground to confirm.

```
ganesha.nfsd -F
```

Client Mounts with Empty Listings

If client mounts show empty listings (and you know that content exists that should be showing), follow these steps:

1. In the Swarm UI, go to Settings > NFS and verify both the export details and the authentication.



Edit export

Name: Accounting Reports

Storage IP(s) or DNS name(s): swarm1.example.com swarm2.example.com

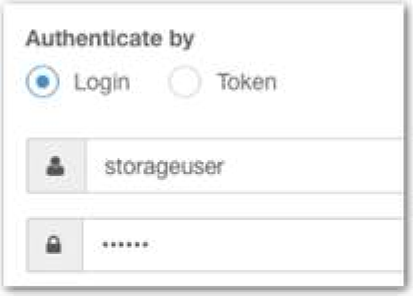
Search host(s): es1.example.com es2.example.com

Search index: hostname-1

Export path: /ACCT_Reports

Scope: mydomain.example.com

acctreports



Authenticate by

● Login ○ Token

storageuser

••••••

2. If that is correct, then, using the configured export details, verify that you can access the bucket via curl from the SwarmNFS server:

```
curl -L -I -v
http://{storageip|name}/{bucket}?domain={storagedomain}  \
 -u {storageuser}:{storageuserpassword}

curl -L -I -v
http://site.example.com/demobucket?domain=storagedomain.example.com
\
 -u storageuser:xxxx
```

3. If you can access the bucket, then verify that you can access Elasticsearch (as defined in the Search host(s) export field) from the SwarmNFS server:

```
curl http://{search host}:9200/_cluster/health -v

curl http://es1.example.com:9200/_cluster/health -v
```

Listing Exports and Clients

Exports — To list active exports from the SwarmNFS server, run the following command:

```
ganesha_mgr show_exports

Show exports
Timestamp:  Thu May 17 17:38:54 2018 76205812  nsecs
Exports:
 Id, path, nfsv3, mnt, nlm4, rquota, nfsv40, nfsv41, nfsv42, 9p, last
 0,  /,  0,  0,  0,  0,  0,  0,  0,  0, Thu May 17 17:38:02 2018,
488596707 nsecs
 2,  /nfsdatadirect,  0,  0,  0,  0,  0,  0,  0,  0, Thu May 17 17:38:02
2018, 488596707 nsecs
 3,  /filefly,  0,  0,  0,  0,  1,  0,  0,  0, Thu May 17 17:38:44 2018,
170782919 nsecs
```

Clients — To list active clients from the SwarmNFS server, run the following command:

```
ganesha_mgr show_client

Show clients
Timestamp:  Thu May 17 17:41:36 2018 780342324  nsecs
Clients:
 IP addr, nfsv3, mnt, nlm4, rquota, nfsv40, nfsv41, nfsv42, 9p, last
 ::ffff:172.30.14.91,  0,  0,  0,  0,  1,  0,  0,  0, Thu May 17
17:38:44 2018 170782919 nsecs
```

Matching Requests between SwarmNFS and Storage Logs

An implementation can have large numbers of unrelated parallel NFS requests. If, for troubleshooting, you need to trace Storage requests back to individual SwarmNFS files that were being read and written, enable verbose (DEBUG) logging and make use of these labels that you can trace through the logs:

- request-type prefix
- fileid
- download/upload id
- part number

> Caution
> If your exports are mounted directly on the SwarmNFS server, do not enable DEBUG logging any longer than necessary.

Missing Custom Header

If a custom header you expect is missing from an object, it may be due to having an invalid name. SwarmNFS skips malformed custom headers silently.

For the rules of custom header naming in Swarm Storage, see Custom Metadata Headers.

Users Lost Permissions

If after a few hours a user becomes unable to read or write files, despite having permissions to do so, you may need to enable session authorization in your SwarmNFS exports.

To have normal reads, writes, and attribute updates go through session authorization, you need to set up superuser access, which is necessary for numerous operations:

- Directory management (create, delete, rename)
- File renaming
- Certain :metadata writes

For 2.1 and higher, this is how to enable session-specific authorization:

1. First, to create session authorization, configure token admin credentials in NFS (user + pass, or token).
2. Next, ensure one of the following:

- In the User Credentials of the NFS export configuration, specify a user that has full access granted by the applicable policy .
- Verify that the token admin as full access granted by the applicable policy.

Caringo Drive

> Optional Swarm Component
> This is an optional Swarm client, with its own distribution packaging and licensing.

Caringo Drive is a virtual drive to make vast Caringo Swarm scale-out object storage accessible by systems running macOS and Windows. Once you install it locally, you can drag and drop files up to Swarm-based cloud storage, and you can view files from Swarm directly on your PC or Mac.

Release Notes

Note the following known issues and watch items:

- With Windows 7, if you cannot Disconnect a Caringo Drive from File Explorer, click Eject Drive in the Caringo Drive application instead.

Supported Operating Systems

Caringo Drive supports the following operating systems:

- Windows Vista and newer (including 64-bit and Windows Server installations)
- macOS (Mac OS X) 10.7 and newer

Preparing the Swarm environment

To support Caringo Drive, make these changes to your Swarm environment:

1. Swarm does not do SSL by default, so install an SSL offloader such as HAProxy.
2. Configure Gateway for S3 and ensure that the S3 bindPort uses the default (80). See Gateway Configuration.
3. Ensure that each Swarm domain to be mapped is reachable via a nameserver. This could be DNS nameserver that supports `*.example.com` notation (such as through a BIND9 file) or via an entry in the `/etc/hosts file`:
   a. Right-click Notepad and choose Run as administrator.
   b. Open the hosts file: `c:\Windows\System32\Drivers\etc\hosts`
   c. Add your entry, such as `172.30.28.109 my.test.domain`

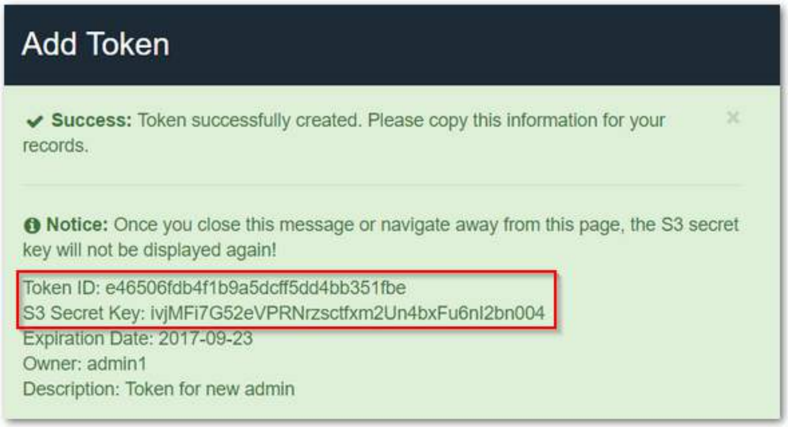
d. Choose File > Save, and reboot to apply the changes.

Creating the S3 tokens

Drives map directly to domains. Before you can set up Caringo Drive on your PC or Mac, you must have an S3 Token for each domain that you want to map via Caringo Drive.

1. From the Content UI, browse to a domain to be mapped and view its Settings (gear icon).
2. Create a token to authorize the domain mapping.
   a. From the toolbar, select Tokens.
   b. Select Add, and enter a helpful Description to identify the token.
   c. Select a duration for the token from the Expiration Date drop-list.
   d. (required) Enable S3 Secret Key, adding a custom (friendly) key string, if desired.
   e. Save the token.
   f. When the Success message appears, copy all token information into a secure document, for reference during drive mapping.

> Important
> You must copy the secret key from the Success message before closing it!



✓ **Success:** Token successfully created. Please copy this information for your records.

Notice: Once you close this message or navigate away from this page, the S3 secret key will not be displayed again!

Token ID: e46506fdb4f1b9a5dcff5dd4bb351fbe
S3 Secret Key: ivjMFi7G52eVPRNrzsctfxm2Un4bxFu6nI2bn004
Expiration Date: 2017-09-23
Owner: admin1
Description: Token for new admin

3. Repeat for each domain for which you need to map a drive.

Installing on macOS

1. Download Caringo Drive for macOS from Caringo Connect (connect.caringo.com).
2. Unpack `CaringoDrive.zip`.
3. Move Caringo Drive.app to the Applications folder.
4. Launch Caringo Drive.
5. Open the Caringo Drive Settings.
6. Create a mapping to a Swarm domain for your Mac by entering the following information:
   - Server: Name of the Caringo domain to be mapped
   - Access Key (Token) ID

- S3 Secret Key

7. Repeat for each additional domain to be mapped.

Installing on Windows

1. (optional) Temporarily disable antivirus software that might interfere with installation.
2. Download Caringo Drive for Windows from Caringo Connect (connect.caringo.com).
3. Run the `CaringoDriveSetup.exe` file.
4. Launch Caringo Drive.
5. Open the Caringo Drive Settings.
6. Create a mapping to a Swarm domain for your PC by entering the following information:
    - Server: Name of the Caringo domain to be mapped
    - Access Key (Token) ID
    - S3 Secret Key
7. Repeat for each additional domain to be mapped.

Usage Notes

- Once a subfolder contains content, it cannot be renamed, due to the S3 API.

Troubleshooting

1. Verify that you have the correct version, which is Caringo Drive v6.3.0 or higher.
2. Locate your logs:
    a. Click the Settings icon.
    b. Select View Logs.
    c. Click Reveal to see the log directory.
3. Important: Enable DEBUG logging.
    a. Edit `/etc/caringo/cloudgateway/logging.cfg`
    b. Change `rootLogger` from `INFO` to `DEBUG`. (see Unexpected HTTP Responses.)
4. Send Caringo Support with your logs for your Gateway(s), including any files that rolled during your usage:
    - /var/log/caringo/cloudgateway_audit.log
    - /var/log/caringo/cloudgateway_server.log
5. Disable DEBUG logging.

Swarm Administration

- Swarm Platform
    - Platform Overview
    - Platform Administration

  - Platform Updates
  - Platform CLI Commands
  - Platform UI
  - Platform Troubleshooting
- Swarm CSN Platform Server
  - CSN Overview
  - CSN Requirements
  - CSN Installation
  - CSN Updates and Upgrades
  - CSN Reset
  - CSN Passwords
  - Scenarios for Swarm UI
  - Installing the Storage UI
  - Migrating Elasticsearch
  - Console - Cluster Services
  - Console - Content Storage
- Swarm Storage Cluster
  - Swarm Concepts
  - Defining Swarm Admins and Users
  - Managing Domains
  - Configuring the Nodes
  - Managing Volumes
  - Using SNMP with Swarm
  - Troubleshooting Storage
  - Managing and Optimizing Feeds
- Elasticsearch for Swarm
  - Snapshot and Restore Search Data
  - Monitoring Elasticsearch
  - Managing Elasticsearch
  - Swarm Historical Metrics
- Swarm Storage UI
  - UI Essentials
  - Viewing and Managing the Cluster
  - Managing Chassis and Drives
  - Viewing and Editing Feeds
  - Using Cluster Settings
  - Configuring SwarmNFS
  - Using Cluster Reports
  - Health Data to Support
  - Legacy Admin Console (port 90)
- Swarm Content Gateway

  - Content Gateway Concepts
  - Content Gateway Authentication
  - Gateway Operations
  - Replicating Domains to Other Clusters
  - Content Metering
  - Gateway Troubleshooting
- Swarm Content UI
  - Content UI Overview
  - Using the Content UI
  - Configuring Tenants
  - Configuring Domains
  - Configuring Buckets
  - Setting Quotas
  - Setting Storage Policies
  - Setting Permissions
  - Setting Tokens
  - Uploading Files
  - Managing Collections
  - Downloading Content
  - Metadata Encoding
  - Editing Names, Metadata, and Versions
  - Usage Reports

Swarm Platform

- Platform Overview
- Platform Administration
- Platform Updates
- Platform CLI Commands
- Platform UI
- Platform Troubleshooting

Platform Overview

The Swarm Platform Server (Platform) unifies and centralizes the services and management information needed to install, upgrade, and monitor Swarm storage clusters. Installed on a dedicated node, Platform simplifies the network services and Caringo product installation process for Swarm administrators.

The Platform infrastructure provides these essential features:

- Required network services. Provides network services that are configured to support a Swarm cluster: DHCP, PXE/Network Boot, TFTP, Syslog, and NTP.
- Service Proxy and Swarm UI. Provides a cluster management dashboard, dynamic configuration of Swarm settings, health report graphics and logs, and Swarm software version management.
- Swarm configuration management. Provides the ability to configure and update the cluster and individual chassis as needed.
- Command-line interface. The CLI gives you a direct interface to Platform management tasks.

Platform Concepts

Services Node for Swarm

The Platform server is an extensible services node that lets you manage Swarm. On a single server, it configures the environment and coordinates the support services needed to deploy hardware into and support a Swarm cluster. It is the next generation of the original Caringo Services Node (CSN).

The Platform server builds the Swarm-centric framework based on Ubuntu MAAS (Metal As A Service), which allows physical servers to be quickly provisioned and collectively managed like virtual machine instances. MAAS manages the machines, and Juju manages the services that run on them. When you bring up a Platform server, it automatically installs and configures the network infrastructure required for the Swarm environment.

Swarm Orchestration

Platform orchestration spans initial Swarm setup and deployment as well as all ongoing maintenance.

- Installation and configuration of Swarm Storage software
- Network boot support
- Automatic provisioning of network and node configs
- Integrated Swarm UI & Service Proxy, for browser-based management

Swarm Environment and Networking

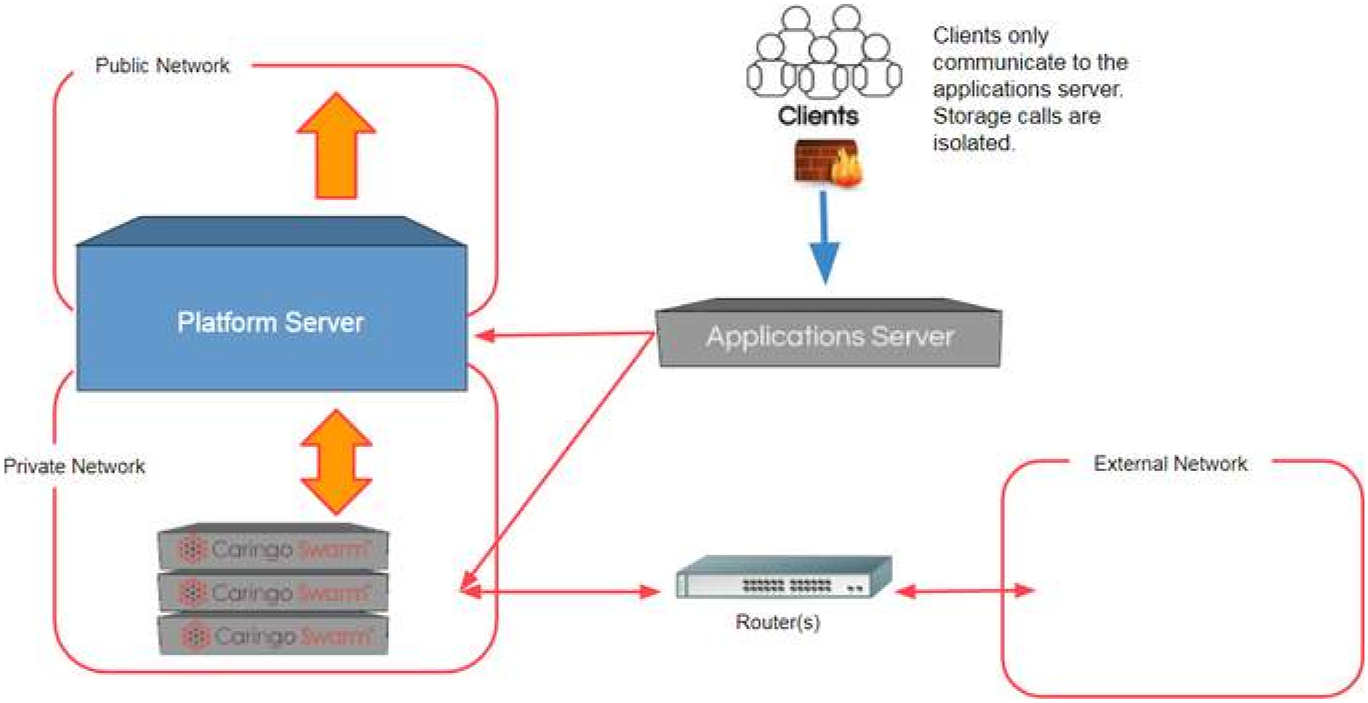
The following system services are set up and initialized by the initial configuration of the Platform server:

1. DHCP server, to allocate Swarm node addresses
2. DNS server, for the management subnet
3. TFTP server, for Swarm network boot
4. NTP time server, essential for Swarm clock synchronization (defaults to pool.ntp.org)
5. Syslog server, for centralized logging
    - Platform and Storage: `/var/log/caringo`
    - Service Proxy: `/var/log/gateway`
6. Service Proxy, a lightweight Gateway instance to host the Swarm UI
7. HTTP proxy, to allow for HTTP access outside the private network
8. Configuration and License server, for Swarm node access

Network Architecture

Swarm architecture uses a private network that is dedicated to the Swarm Storage nodes and application servers. Access to this cluster is restricted to the Platform server and any applications that reside on private subnet (in the absence of existing routers). Platform server owns this private storage network, but it can support flexible network topologies.

- Platform server addresses are IPs on a routable subnet.
- Application servers can be dual-homed like the Platform server, for ease of access to cluster.
- Swarm Storage nodes' default gateway is the Platform server, which can be changed to support existing routers.



Swarm Interfaces

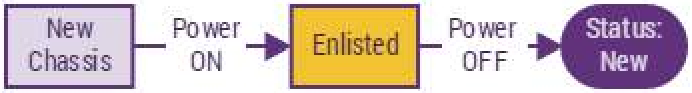
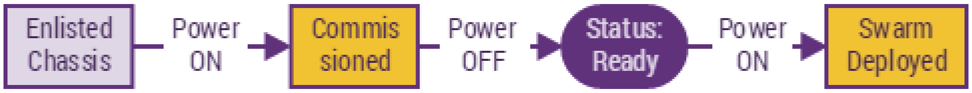
There are three interfaces to Platform functionality:

- CLI (Command-Line Interface) — see Platform CLI Commands
  - The CLI is a native application installed on the Platform server. Versions for other platforms are available, so it can run anywhere.
  - To get CLI help, just type: `platform`
  - All CLI commands start with “platform”:
    - To list all nodes: `platform list nodes`
    - To deploy a node: `platform deploy storage`
- Swarm UI — see Platform UI
- REST API, underlying both the CLI and the Swarm UI
  - API is discoverable through the root: `http://<platform>:8095/platform`

Chassis Lifecycle

Each chassis goes through the following lifecycle on the way to becoming a functioning Swarm Storage node. Once a chassis is running as a Storage node, it will always boot as a Storage node and will have a statically assigned IP address that is consistent across reboots.

> Note
> The Platform Server uses IPMI for power control, so the IPMI port must be accessible from the Platform Server.

1.	Add Hardware Enlist	Before new hardware can be managed by the Platform server, you need to power it on manually for the first and only time. This initial step enlists the machine with the Platform server, after which the hardware will automatically power itself off. After enlistment, new hardware appears in the CLI with the state "New", and it can remain in that state indefinitely until needed. Note: Enlistment is non-destructive to existing data. If the machine was added by mistake to the subnet that Platform server manages and is enlisted, you can correct it. No installation happens until you use the CLI to deploy Swarm to the chassis.  New Chassis Power ON Enlisted Power OFF Status: New
2.	Deploy Hardware Commission	Once new hardware is enlisted, Platform sees it as available for commissioning, which occurs when you run a CLI command that deploys the hardware and boots Swarm. The deployment command automatically pushes the new hardware through all of the required operations. After the deploy operation starts, you can use the "list nodes" CLI command to view the chassis moving through the different states. Note: The first power on and commissioning steps are also non-destructive to any existing data on the disks. Any data on the disks will persist until Swarm is booted on the second power on operation, after which the persistence of any existing data is determined by Swarm.  Enlisted Chassis Power ON Commis sioned Power OFF Status: Ready Power ON Swarm Deployed

Chassis Statuses

As a chassis moves through the phases of deployment, it may show the following statuses:

- New — It has been powered on for the first time and has enlisted with the Platform Server.
- Commissioning — It is in the process of or just finished the initial hardware interrogation stage on the way to deployment.
- Ready — It has completed all the initial stages and is ready to be deployed with software.
- Deploying/Deployed — It is in the process of or finished with the deployment of the Storage software.
- Failed <X> — It encountered an error in completing one of the stages.

Chassis Grouping

If you make no explicit subcluster assignments in the Swarm configuration and there are more than one storage process running on a chassis, then each chassis forms a de facto subcluster. In Swarm, the "node.subcluster" value is an free-form name that you assign to one or more chassis.

The storage process looks at all the names assigned to the different chassis and forms them into groups, which can then be used to determine how replicas are handled. You can group the chassis in any way you need to achieve your desired replica/fail-over paradigm.

Platform Administration

Platform's CLI (command-line interface) is installed by default on the Platform server and supports common administrative tasks.

- Rebooting a Cluster
- Adding a Chassis
- Reconfiguring the Cluster
- Reconfiguring a Chassis
- Releasing a Chassis
- Resetting to Defaults
- Managing Subclusters
- Changing the Default Gateway
- Managing Administrators
- Upgrading Swarm Storage
- Managing Service Proxy
- Configuring DNS
- Configuring Docker Bridge

Rebooting a Cluster

There are two CLI options for rebooting a cluster: full versus rolling.

- Full reboot notifies every chassis in the cluster to reboot itself at the same time, meaning that the entire cluster will temporarily be offline as the chassis reboot.

  Full reboot

  ```
  platform restart storagecluster --full
  ```

- Rolling reboot is a long-running process that keeps the cluster operational by rebooting the cluster one chassis at a time, until the entire cluster has been rebooted. A rolling reboot includes several options, such as to limit the reboot to one or more chassis:

Rolling reboot

```
platform restart storagecluster --rolling
 [--chassis <comma-separated system IDs>]
 [--skipConnectionTest]
 [--skipUptimeTest]
 [--continueWithOfflineChassis]
 [--stopOnNodeError]
```

Requirements

Before a rolling reboot can begin, these conditions must be met:

1. All chassis targeted for rebooting must be running and reachable. If you have offline chassis, be sure to set a flag to have them ignored:
   - To skip the connection check altogether, add the flag `--skipConnectionTest`
   - To have the reboot process ignore currently offline chassis, add the flag `--continueWithOfflineChassis`
2. All chassis must have an uptime greater than 30 minutes. To skip this requirement, add the flag `--skipUptimeTest`

Managing Rolling Reboots

You have 10 seconds to cancel a rolling reboot before it begins. Once a rolling reboot has started, it will stop and report an error if any of these occur:

1. A chassis is offline when it is selected for reboot. To have the reboot process ignore currently offline chassis, add the flag `--continueWithOfflineChassis`.
2. The reboot process will continue if the volumes come up but a node goes into an error state. To have the reboot process stop, add the flag `--stopOnNodeError`.
3. If the chassis boots with a number of volumes that doesn't match the number present before the chassis was rebooted. A volume is considered up if it has a state of: ok, retiring, retired, or unavailable
4. The chassis does not come back online after 3 hours has passed.

If a rolling reboot has stopped due to an error, you can resume the reboot using the `resume` command below after you have resolved the error.

Status check — To retrieve the status of a rolling reboot task, use the following commands for reboots remaining and reboots completed:

Rolling reboots remaining

```
platform status rollingreboot
```

Rolling reboots completed

```
platform status rollingreboot --completed
```

Global states — When viewing the status for a rolling reboot, a rolling reboot task can have the following global states:

- in-progress: The rolling reboot is currently running.
- paused: The rolling reboot has been paused (using the `pause` command).
- completed: The rolling reboot finished successfully.
- cancelled: The rolling reboot was caused per a user request.
- error: The reboot has been stopped due to an error of some kind.

Chassis states — The status listing will also show the status for each chassis that is processed by the rolling reboot task. Each chassis can have one of the following states:

- pending: The rolling reboot task has yet to process the chassis.
- in-progress: The rolling reboot task is in the process of rebooting the chassis.
- completed: The chassis was successfully rebooted.
- removed: The chassis was removed from the list of chassis to process after the rolling reboot was started (using the `delete rolling reboot` command).
- error: The chassis encountered an error of some kind.
- abandoned: The chassis was currently being processed when a user cancelled the rolling reboot.
- dropped: The rolling reboot was in the process of waiting for the chassis to reboot when a user request was made to move to the next chassis (using the `--skip` flag).
- offline: The chassis was already offline when the reboot task attempted to reboot the chassis.

Cancel reboot — To cancel (not pause) an active rolling reboot, issue the delete command, which the reboot process at the earliest moment and thus cannot be restarted later.

```
platform delete rollingreboot --cancel
```

Exclude from reboot — To exclude from a currently running rolling reboot one or more chassis that have not yet been rebooted:

```
platform delete rollingreboot --chassis <comma-separated system IDs>
```

Pause reboot — To pause the current rolling reboot process so that it can be restarted later:

```
platform pause rollingreboot
```

Resume reboot — To resume a paused rolling reboot:

```
platform resume rollingreboot
```

No-wait reboot — Normally, the rolling reboot process will wait up to 3 hours for a rebooted chassis to come back online before proceeds to the next. To force the process to stop waiting and move to the next chassis, use the `--skip` flag:

```
platform resume rollingreboot --skip
```

Adding a Chassis

Which version of Swarm a given node uses is set at the time of provisioning.

To add a single chassis as a new Swarm node, use the following process:

1. Create a node.cfg file and add any node-specific Swarm settings to apply, or leave it blank to accept all current settings.
2. Power on the chassis for the first time.
3. Wait until the chassis enlists and powers off.
4. Deploy the new server:

    ```
    platform deploy storage -n 1 -v <#.#.#-version-to-deploy>
    ```

To deploy an individual chassis by system ID, use this process:

1. Create a node.cfg file and add any node-specific Swarm settings to apply, or leave it blank to accept all current settings.
2. Get a list of chassis that are available for deployment by using the following command:

    ```
    platform list nodes --state New
    ```

3. Choose a System ID to deploy a single chassis using a command like the following:

```
platform deploy storage -y 4y3h7p -v 9.2.1
```

Service Proxy

If the Service Proxy is running on the Platform Server when you add or remove chassis, be sure to restart the service so that it can pick up the new chassis list:

```
platform restart proxy
```

Reconfiguring the Cluster

You can modify your cluster-wide Swarm configuration at anytime using the CLI and a configuration file. The reconfiguration process is additive: all existing settings that are not referenced in your file are preserved. That is, if you define only two settings, Platform overwrites or adds only those two settings.

1. Create a supplemental .cfg file (such as `changes.cfg`) and specify any new or changed Swarm settings to apply.
2. To upload your configuration changes, use the following CLI command:

```
platform upload config -c {Path to .cfg}
```

The CLI will parse the uploaded configuration file for changes to make to Platform.

If Swarm was running during the upload, Platform Server attempts to communicate the new configuration to Swarm. Any settings that cannot be communicated to Swarm will require a reboot of the Swarm cluster in order to take effect. For each setting contained in the file, the CLI will indicate if the setting was communicated to the Storage cluster and if a reboot is required. The Swarm UI also indicates which settings require rebooting.

Example: Increase Swarm processes

> Swarm 10
>
> Swarm Storage 10 has a single-process architecture, so the configuration setting `chassis.processes` is no longer used and cannot be increased.

Option 1: Create a configuration file:

To set all chassis throughout the cluster to a higher number of processes, you would create a configuration file and upload it to Platform Server.

1. Create a text file, such as `update.cfg`, containing only the setting to be changed.

```
chassis.processes = 6
```

2. To upload your configuration changes, use the following CLI command:

```
platform upload config -c {Path to update.cfg}
```

> Note
> Include the `-m <mac-address>` parameter if you want to target the update to specific chassis.

Option 2: Use the CLI directly:

1. Add the configuration change directly:

```
platform add config --name "chassis.processes" --value 6
```

Reconfiguring a Chassis

You can modify the node-specific settings for a single chassis by the same process, but you need to specify the MAC address of any valid NIC on that chassis.

1. Create a .cfg file (such as `changes.cfg`) and specify any new or changed node-specific settings to apply.
2. To upload your configuration changes, use the following CLI command:

```
platform upload config -c {Path to .cfg} -m {mac address}
```

The CLI will parse the uploaded configuration file for changes to make to that chassis.

Releasing a Chassis

There may be times when you need to release a chassis from the Swarm cluster, either for temporary maintenance or for permanent removal.

> Important
> To ensure a clean shut down, power off the chassis through the UI or SNMP before you run `release` commands.

Temporary release — Temporary release of a chassis assumes that the chassis will be added back into the cluster at a later time. Releasing a chassis lets you unallocate its cluster resources, such as IP Addresses, or wipe and reset its configuration.

Once the chassis is powered off, you can release the chassis from the Swarm cluster:

Temporary removal

```
platform release storagechassis -y <system-id>
```

Permanent removal — Permanent removal is for retiring a chassis altogether or changing the chassis' main identifying information, such as changing a NIC. Removing the chassis from management will cause the chassis to start the provisioning life cycle as if it were a brand new chassis, if it is powered on again.

Once the chassis is powered off, you can remove the chassis from Platform Server management permanently:

Permanent removal

```
platform release storagechassis -y <system-id> --remove
```

Resetting to Defaults

If you would like to clear out all existing setting customizations from a given chassis or the entire cluster, you can issue the following commands.

> Note
>
> These commands require a cluster reboot because the reset is not communicated to the Storage network dynamically.

Delete All Default Chassis Settings

```
platform delete allchassisconfig
```

Delete All Cluster Settings

```
platform delete allclusterconfig
```

Managing Subclusters

After all the chassis have been deployed and are running, you can assign chassis to subclusters. To see the current subcluster assignments, use the `list` command:

List subclusters

```
platform subcluster list
```

To assign a chassis to a subcluster, use the `assign` command:

Add to subcluster

```
platform subcluster assign -y <system-id> --subcluster <subcluster-name>
```

Note
Reassignment is not immediate. Allow time for every node on the chassis to be migrated to the new subcluster.

To remove a chassis from a subcluster, use the `unassign` command:

Remove from subcluster

```
platform subcluster unassign -y <system-id>
```

Important
Reboot the chassis for the subcluster removal to take effect.

Changing the Default Gateway

By default, the Platform Server configures Swarm Storage to use the Platform Server as its default gateway. To override this behavior, either add a "network.gateway" to your cluster configuration file or issue the following command:

```
platform add config --name "network.gateway" --value "<ip-of-gateway>"
```

Managing Administrators

With one exception, modifying the admin users for the Storage cluster requires the Storage cluster to be up and running before the operations can be done. The one exception to this is the "snmp" user which which can have its password set while the cluster is down or before the cluster has been booted for the first time.

> Important
> Changing the password for the "snmp" user requires a full cluster reboot for the change to take effect.

Adding or Updating Users

> Important
> Modifying passwords for the admin user will require you to restart the Service Proxy, if you have it installed. It could also require updates to Gateway configuration.

To add a new admin user, use the following CLI command

Add admin user

```
platform add adminuser
 [--askpassword]
 [--username <username>]
 [--password <user password>]
 [--update]
```

The `--askpassword` flag lets you avoid specifying a password via the command line by providing the password via stdin. When this flag is used, you'll be prompted to enter a new/updated password for the user. You can also use the Linux pipe functionality:

```
cat password.txt | platform add adminuser --askpassword --username admin
--update
```

> Important
> If you are just updating the password for an existing user, be sure to use the `--update` flag.

To delete an admin user from the cluster, use the following CLI command:

Delete admin user

```
platform delete adminuser --username <username>
```

Upgrading Swarm Storage

To upgrade Swarm Storage in a live cluster, you can use the CLI to upload the version and then take steps to deploy it to your running nodes, either by restarting the entire cluster or each chassis in turn.

> Note
> The `deploy storage --upgrade` command is used for both upgrades and downgrades of Storage versions.

1. Upload the new version of the Swarm Storage software to Platform server, making sure that the <version-name> matches the version of Swarm Storage being uploaded:

   ```
   platform upload storageimages -i <path-to-zip> -v <version-name>

   platform upload storageimages -i ./storage-9.6.0-x86_64.zip -v 9.6
   ```

   > Note
   > The zip file to use above is contained with the `Swarm-{version}-{date}.zip` file that was downloaded. Inside this zip, there is a folder called Storage which contains a file called `storage-{version}-x86_64.zip`. This is the zip file to use for the command above.

2. Get a full listing of all of the nodes and their IPs, MAC addresses, and system IDs:

   ```
   platform list nodes --state Deployed --short
   ```

3. Using the list of system IDs, deploy the upgrade on each of the nodes. If you want to restart the node immediately after upgrade, run that command as well:

   ```
   platform deploy storage --upgrade -v 9.2.1 -y <system-id>
   platform restart storagenode -y <system-id>
   ```

4. If you did not restart each node individually, restart the cluster now, either full or rolling:

   ```
   platform restart storagecluster --full
   or
   platform restart storagecluster --rolling [<options>]
   ```

Managing Service Proxy

Status — To check the status of the Service Proxy, use this command:

```
platform status proxy
```

Upgrade — To upgrade the Service Proxy on the Platform server, use the CLI to upload the version and deploy it:

```
platform deploy proxy -b <path-to-zip> --upgrade
```

> Note
> After a Service Proxy upgrade, it will take several minutes for the UI to come back up.

Configuring DNS

You may need to have the Storage nodes resolve names for outside resources, such as Elasticsearch or Syslog. To do so, configure the DNS server on the Platform Server to communicate with outside domains.

Option 1: Simple forwarding

A Slave/Backup DNS zone is a read-only copy of the DNS records; it can only receive updates from the Master zone of the DNS server.

If you have no DNS master/slave relationships configured, you can do simple forwarding by having the domain managed by the Platform server forward all lookups to outside domains:

1. Edit `/etc/bind/named.conf.options` and add the following line after the "`listen-on-v6`" line

   ```
   forwarders {172.30.0.202;};
   ```

2. Run the following command to restart bind9 on the Platform Server:

```
sudo systemctl restart bind9
```

Option 2: Configuring a Slave DNS Zone

If you have an external DNS Zone configured, you can have the Platform Server become a slave DNS of that zone; the reverse can be done to allow other systems to resolve names for servers managed by the Platform server.

This process assumes that the external DNS server has been configured to allow zone transfers to the Platform server. The DNS server on the Platform server is not configured to restrict zone transfers to other DNS slaves.

1. Edit `/etc/bind/named.conf.local` and add the following line at this location:

```
// slave other local zones
include "/etc/bind/named.conf.slaves";
```

2. Create a new file called `/etc/bind/named.conf.slaves` and add your settings in this format:

```
// local slave zones
zone "example.com" in {
    type slave;
    masters {172.30.0.100; };
    file "/var/cache/bind/slave/zone-example.com";
};
```

3. Run the following command to restart bind9 on the Platform Server:

```
sudo systemctl restart bind9
```

Configuring Docker Bridge

To configure or modify the network information that is used by the default Docker (docker0) bridge, edit the file `/etc/docker/daemon.json`. You can add networking properties as properties to the root JSON object in the file:

```
{
  "log-opts": {
    "max-size": "5m",
    "max-file": "10"
  },
  "bip": "10.0.1.1/24"
}
```

The `bip` property sets the IP address and subnet mask to use for the default docker0 bridge. For details on the different properties, see the Docker documentation.

Platform Updates

To upgrade the Platform Server:

1. Download the Swarm bundle ZIP and locate the `Platform` folder within it.
2. Transfer the ZIP file containing the new version of the Platform server.
3. Unzip the file and cd into the new directory.
4. Run the following command:

   ```
   sudo ./upgradeplatform.sh
   ```

> Note
> When upgrading from 9.0 to 9.1, the Platform server will download about 400MB in updates from the Internet. As such, the total upgrade time will vary depending on bandwidth.

After updating Platform to a newer version, update the components that you use with it as needed:

1. Swarm Storage (upload the storage ZIP from the bundle) — see Upgrading Swarm Storage in Platform Administration.
2. Swarm UI, depending on your implementation:
   - Service Proxy (includes Swarm UI in its ZIP) — see Managing Service Proxy in Platform Administration.
   - Standalone Content Gateway that hosts Swarm UI — see the Implementing Content Gateway section in the Release Notes and Upgrading the Storage UI in Swarm Storage UI Installation.

Platform CLI Commands

The Swarm Platform configures and manages Swarm storage clusters. This command line interface helps you automate your hardware management tasks by letting you script common, high-level management tasks around cluster and node deployment:

Cluster	Create a storage cluster Read a storage cluster's settings Update a storage cluster's settings
Chassis	Add a chassis to the cluster Read a chassis' settings Update a chassis' settings
Software	Upgrade a storage cluster's software Add or update a storage license Update the storage cluster configuration, such as the location of Syslog and NTP servers

CLI Commands

- platform
- platform add
- platform add adminuser
- platform add bonding-mode
- platform add config
- platform add iprange
- platform add kernelparam
- platform add tag
- platform bootstrap
- platform delete
- platform delete adminuser
- platform delete allchassisconfig
- platform delete allclusterconfig
- platform delete bonding-mode
- platform delete config
- platform delete iprange
- platform delete kernelparam
- platform delete proxyimage
- platform delete rollingreboot
- platform delete storageimages
- platform delete tag
- platform deploy
- platform deploy proxy
- platform deploy storage
- platform list
- platform list adminusers
- platform list bonding-mode
- platform list config

- platform list license
- platform list nodes
- platform list proxyimages
- platform list storageimages
- platform list subnets
- platform list tags
- platform pause
- platform pause rollingreboot
- platform release
- platform release proxy
- platform release storagechassis
- platform restart
- platform restart proxy
- platform restart storagecluster
- platform restart storagenode
- platform resume
- platform resume rollingreboot
- platform status
- platform status platform
- platform status proxy
- platform status rollingreboot
- platform status storagechassis
- platform status storagecluster
- platform subcluster
- platform subcluster assign
- platform subcluster list
- platform subcluster unassign
- platform upload
- platform upload config
- platform upload license
- platform upload storageimages

platform

A CLI for the Platform node

A CLI for the Platform node

The Platform CLI provides a command line interface to the REST API

```
platform
```

Options

```
--port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
      --version         Print version info
```

platform add

Add to the Platform node

Add to the Platform node using add subcommands

```
platform add

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform add adminuser

Add an admin user to the Storage cluster

Add a new admin user to the storage cluster. Can also be used to update the password for an existing admin user.

When updating the password for the snmp user, a cluster restart is required for the change to take effect. The cluster does not need to be running in order to change the password.

When updating the password for any non-snmp user, the cluster must be running in order to the operation to succeed.

```
platform add adminuser
```

Options

```
--askpassword       Use to read password from stdin (optional)
      --password string   The password to use
      --update            Use to just update the password for the given user (optional)
      --username string   The username to add

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform add bonding-mode

Set default bonding mode

Sets the default bonding mode to use when booting Storage chassis.

```
platform add bonding-mode
```

Options

```
--default-bonding-mode string   The default bonding mode

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform add config

Upload chassis config

Upload a chassis configuration file to the Platform server. For example:

./platform upload config -c node10net.cfg

```
platform add config
```

Options

```
-m, --mac string     The MAC address for a specific storage chassis (optional)
      --name string    The name of the setting
      --override       Used to set custom configuration settings (optional)
      --value string   The value of the setting

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform add iprange

Add a reserved IP range

Add a new reserved range of IP addresses to a DHCP subnet. For example:

./platform add iprange -a 172.16.1.1 -z 172.16.1.30

```
platform add iprange
```

Options

```
-z, --endip string     The ending IP of the reserved range to use
  -a, --startip string   The starting IP of the reserved range to use

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform add kernelparam

Add a kernel parameter to a chassis

Add a kernel parameter to a chassis. Requires a reboot of the chassis.

```
platform add kernelparam
```

Options

```
--kernel_params string   The kernel parameters to add
      --replace                Use to replace and not just append kernel parameters (optional))
  -y, --systemid string        The system_id of the chassis

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform add tag

Add a node tag

Add a tag to a chassis

```
platform add tag
```

Options

```
-y, --systemid string   The system_id of the chassis
      --tagName string    The name of the tag go add

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform bootstrap

Prepare the Platform environment services for Storage node deployment

Bootstrap the Platform services, adding Storage images and prepare the Swarm cluster environment.

Bootstrapping the Platform server uploads a starting Storage image, identifies the subnet for DHCP and prepares the Platform server to support the Storage chassis with enlisting/commissioning images while creating settings needed for the boot of chassis to a chosen image. For example:

./platform bootstrap -t 10.0.0.0/24 -a 10.0.0.100 -z 10.0.0.200 -i ./Swarm-v9.1.1-x86_64.zip -v 9.1.1

```
platform bootstrap
```

Options

```
-s, --dnssuffix string        The default DNS domain suffix to use
  -z, --endip string            The ending IP of the DHCP range to use
      --force                   Use to force an unrecommended dynamic range
  -i, --image string            The full path for the Storage image
      --offline                 Use for offline installations
  -a, --startip string          The starting IP of the DHCP range to use
  -v, --storageversion string   The version of Storage image to use
  -t, --subnet string           The subnet to use for DHCP

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete

Delete command

Delete objects using the subcommands

```
platform delete

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete adminuser

Delete an admin user to the Storage cluster

Delete a new admin user to the storage cluster

```
platform delete adminuser
```

Options

```
--override          Use to delete user even if Storage can not be notified of change. Use
caution. (optional)
      --username string     The username to delete

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete allchassisconfig

Delete all chassis config settings

Clear all chassis configuration settings. This will reset all settings back to their default.

NOTE: A cluster or chassis reboot is needed after this command is run.

```
platform delete allchassisconfig
```

Options

```
--force        Use to not prompt for verification
  -m, --mac string   The MAC address for a specific storage node (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete allclusterconfig

Delete all cluster config settings

Clear all cluster configuration settings. This will reset all settings back to their default.

NOTE: A cluster reboot is needed after this command is run.

```
platform delete allclusterconfig
```

Options

```
--force   Use to not prompt for verification

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete bonding-mode

Delete default bonding mode

Delete the default bonding mode to use when booting Storage chassis.

```
platform delete bonding-mode

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete config

List node configuration

list config will display the configuration settings for the Caringo Storage cluster or for a single Storage chassis. For example:

$> ./platform list config

```
platform delete config
```

Options

```
-m, --mac string            The MAC address for a specific storage node (optional)
      --propertyname string   The name of the property to delete

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete iprange

Delete a reserved IP range

Delete a reserved range of IP addresses. For example:

```
./platform delete iprange -a 172.16.1.1 -z 172.16.1.30
```

```
platform delete iprange
```

Options

```
-a, --startip string   The starting IP of the reserved range to use

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete kernelparam

Delete kernel parameters for a chassis

Delete kernel parameters for a chassis. Requires a reboot of the chassis.

```
platform delete kernelparam
```

Options

```
-y, --systemid string   The system_id of the chassis

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete proxyimage

Delete Service Proxy image

Delete Service Proxy software image:

```
platform delete proxyimage
```

Options

```
--version string   The version of the Service Proxy to remove

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete rollingreboot

Cancel rolling reboot

Cancel a currently running rolling reboot.

This command can also be used to remove chassis from a currently running rolling reboot using the --chassis argument

```
platform delete rollingreboot
```

Options

```
--cancel           Do a full cancel of the rolling reboot
      --chassis string   List of chassis to remove from rolling reboot. A comma-separated list of system IDs (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete storageimages

Delete a Storage image

Delete a Caringo Storage image from the Platform server. For example:

./platform delete storageimages -v 9.1base

```
platform delete storageimages
```

Options

```
-v, --storageversion string   The version of Storage image to delete

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform delete tag

Delete a node tag

Delete a tag to a chassis

```
platform delete tag
```

Options

```
-y, --systemid string   The system_id of the chassis
      --tagName string    The name of the tag go add

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform deploy

Deploy command

Deploy command

```
platform deploy

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform deploy proxy

Deploy Service Proxy

Deploy a local Service Proxy instance:

Deploy a new Service Proxy instance, supplying the zip file for upload. The Storage cluster will be validated, ensuring there are at least 2 Storage nodes running with the cluster settings for enforceTenancy on and noauth on.

For example:

./platform deploy gateway -b ./service_proxy.zip

```
platform deploy proxy
```

Options

```
-b, --proxybundle string   The full path for the Service Proxy bundle zip file
      --upgrade              upgrade
      --version string       The version of the Service Proxy to use

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform deploy storage

Deploy Storage chassis

Deploy Swarm Storage chassis with a selected Storage image.

Deploy Storage chassis by supplying the configuration file and number of chassis. Optionally deploy to a single chassis by specifying its systemid instead of a number of chassis. For example:

./platform deploy storage -n 1 -c ./node10net.cfg -v 9.1.1

```
platform deploy storage
```

Options

```
--bondingMode string      The network bonding mode to use for the chassis (optional)
      --kernelparams string     general kernel params for the chassis (optional)
      --manual                  Force the chassis you use manual power control (optional)
  -n, --numchassis int          The number of chassis to deploy
  -c, --storageconfig string    The full path for the Storage configuration (node.cfg) file
  -v, --storageversion string   The version of Storage image to use
  -y, --systemid string         The system_id of the chassis to deploy to (optional)
      --upgrade                 Just do an upgrade operation (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list

List configurations and objects

List configurations and objects in Platform

```
platform list

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list adminusers

List admin users

List the current admin users for the Storage cluster

```
platform list adminusers
```

Options

```
--validate   Validate the current passwords with Storage nodes to make sure they match
(optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list bonding-mode

Show default bonding mode

Show the default bonding mode to use when booting Storage chassis.

```
platform list bonding-mode

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list config

List node configuration

list config will display the configuration settings for the Caringo Storage cluster or for a single Storage chassis. For example:

$> ./platform list config

```
platform list config
```

Options

```
--all          List all properties (optional)
  -m, --mac string   The MAC address for a specific storage node (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list license

Get current license

Show the current license for storage cluster

```
platform list license
```

Options

```
--plain   Use to list the license as plain text

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list nodes

Get list of chassis

Shows a list of chassis that are currently being managed by Platform.

Each line of output represents a single chassis being managed by Platform. For each chassis the properties will be displayed in a space delimited list of key-value pairs. For example:

./platform list nodes

```
platform list nodes
```

Options

```
--dumpall            Set to dump all the node properties
      --long               Long format
      --only-system-id     Use to only show System IDs (optional)
      --os string          Only show nodes that match a given OS (optional)
      --osversion string   Only show nodes that match a given OS version (optional)
      --state string       Only show nodes that match a given state (optional)
  -y, --systemid string    The system_id of the chassis to list information for (optional)
      --tag string         Only show nodes that match a given tag (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list proxyimages

List available Service Proxy images

Return the list of Service Proxy images available for deploying. For example:

$> ./platform list proxyimages

Available Service Proxy image: 9.0.0

```
platform list proxyimages

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list storageimages

List available Storage images

Return the list of Caringo Storage boot image available from the Platform node for deploying. For example:

$> ./platform list storageimages

Available Storage image: 9.0.0

```
platform list storageimages

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list subnets

List subnets

List the known subnets and the reserved IP ranges for each. For example:

./platform list subnets

```
platform list subnets
```

Options

```
--all   Show all available subnets

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform list tags

List node tags

list all the MAAS tags assigned to chassis

```
platform list tags
```

Options

```
-y, --systemid string   The system_id of the chassis

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform pause

Pause

Pause command

```
platform pause

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform pause rollingreboot

Pause rolling reboot

Pause a rolling reboot. The rolling reboot will finish processing the current chassis and then pause. It can be restarted later.

```
platform pause rollingreboot

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform release

Release

Release command

```
platform release

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform release proxy

Release Service Proxy

Release the local Service Proxy

```
platform release proxy

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform release storagechassis

Release chassis

Release Swarm Storage chassis.

Release Storage chassis by supplying the system id of the chassis.

```
platform release storagechassis
```

Options

```
--remove            Use flag to delete the chassis altogether
  -y, --systemid string   The system_id of the chassis to release

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform restart

Restart

Restart command

```
platform restart

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform restart proxy

Restart Service Proxy

Restart the local Service Proxy.

A restart is useful after a settings change. This subcommand will restart the local Service Proxy service. For example: ./platform restart proxy

```
platform restart proxy

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform restart storagecluster

Restart cluster

Restart the Storage cluster.

A restart is useful after a settings change. This subcommand will restart the Storage cluster. For example: ./platform restart storagecluster

```
platform restart storagecluster
```

Options

```
--chassis string               List of chassis to include in a rolling reboot. A
comma-separated list of system IDs (optional)
      --continueWithOfflineChassis   Use to continue with rolling reboot if a chassis is
offline when reboot is attempted (optional)
      --full                         Full reboot
      --rolling                      Rolling reboot
      --skipConnectionTest           Use to skip a test to make sure every chassis is up and
reachable (optional)
      --skipUptimeTest               Use to skip a test to make sure every chassis has been up
more than 30 minutes (optional)
      --stopOnNodeError              Use to stop rolling reboot if node boots into an error
state (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform restart storagenode

Restart node

Restart a Storage node.

```
platform restart storagenode
```

Options

```
-y, --systemid string   The system_id of the chassis to restart

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform resume

Resume

Resume command

```
platform resume

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform resume rollingreboot

Resume rolling reboot
Resume a rolling reboot.

```
platform resume rollingreboot
```

Options

```
--continueWithOfflineChassis   Use to continue with rolling reboot if a chassis is offline when
reboot is attempted (optional)
      --skip                        Skip waiting for the current chassis to come back up and
move to the next.
      --stopOnNodeError             Use to stop rolling reboot if node boots into an error
state (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform status

Status
Status command

```
platform status

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform status platform

Platform Status
Get status for Platform server

```
platform status platform

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform status proxy

Service Proxy Status
Get status for Service Proxy

```
platform status proxy
```

Options

```
--short   Short format

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform status rollingreboot

Rolling Reboot Status

Get status for a running rolling reboot

```
platform status rollingreboot
```

Options

```
--completed   Use to show completed jobs

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform status storagechassis

Chassis Status

Get status for a Storage chassis

```
platform status storagechassis
```

Options

```
-y, --systemid string   The system_id of the chassis to list status for

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform status storagecluster

Cluster Status

Get status for Storage Cluster

```
platform status storagecluster

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform subcluster

Perform subcluster assignment operations

Perform subcluster assignment operations

```
platform subcluster

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform subcluster assign

Assign subcluster

Assign a chassis to a subcluster

```
platform subcluster assign
```

Options

```
--subcluster string   The name of the subcluster to assign the node to.
  -y, --systemid string     The system_id of the chassis to assign

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform subcluster list

Used to get a list of subcluster assignments
Shows a list of subclusters and their chassis assignments.

```
platform subcluster list
```

```
      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform subcluster unassign

Unassign subcluster
Unassign a chassis to a subcluster

```
platform subcluster unassign
```

Options

```
-y, --systemid string   The system_id of the chassis to unassign

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform upload

Upload to Platform
Upload configs and binaries to Platform

```
platform upload
```

```
      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform upload config

Upload chassis config
Upload a chassis configuration file to the Platform server. For example:
./platform upload config -c node10net.cfg

```
platform upload config
```

Options

```
-c, --config-file string   The full path for the storage configuration file
  -m, --mac string           The MAC address for a specific storage chassis (optional)
      --override             Used to set custom configuration settings (optional)

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

platform upload license

Upload license to Platform
Add a Caringo Storage license file to the Platform node. For example:
./platform upload license --file "/home/admin/license.txt"

```
platform upload license
```

Options

```
-f, --file string   The full path for the Storage license file

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1"
```

platform upload storageimages

Upload image to Platform
Add a new Caringo Storage boot image to the Platform server for deploying Storage chassis. For example:
./platform upload storageimages --image "/home/admin/image.zip" --storageversion "9.2.0"

```
platform upload storageimages
```

Options

```
-i, --image string            The full path for the Storage image
  -v, --storageversion string   The version of Storage image to use

      --port int        Platform server node API port (default 8095)
      --server string   Platform server node IP address (default "127.0.0.1")
```

Platform UI

When Platform is running, it augments the Swarm UI with additional pages and capabilities.

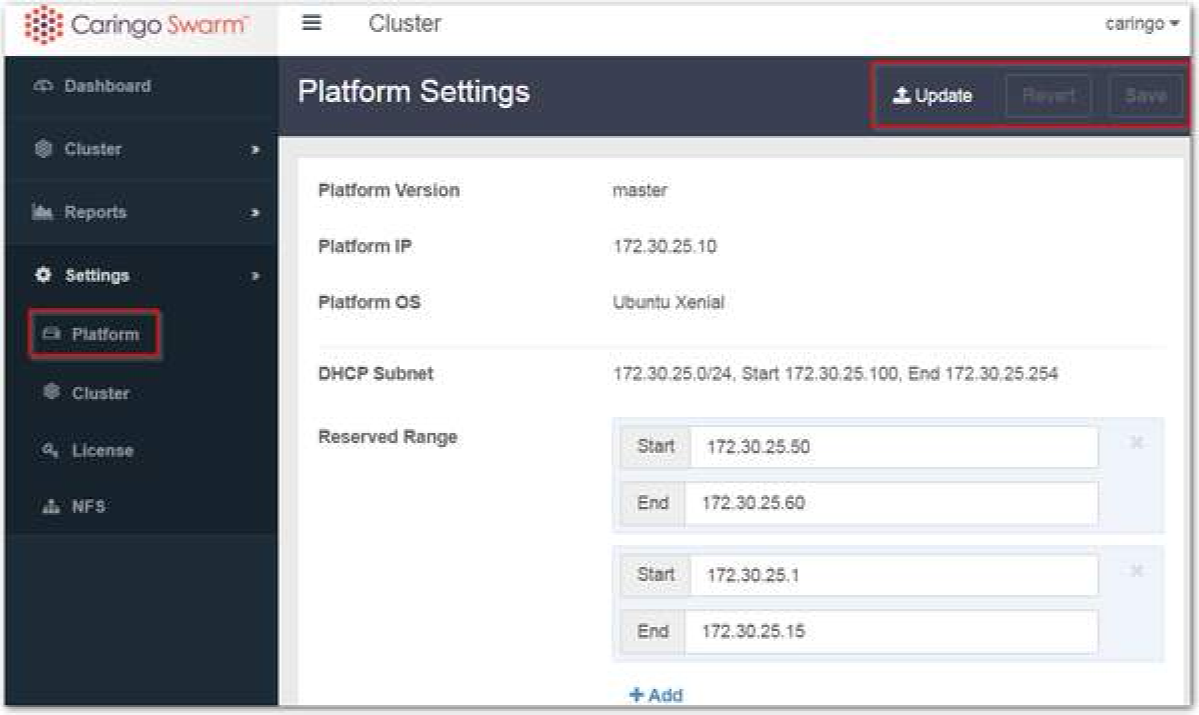
- Platform Settings
- Chassis Settings
- Restarting Hardware

Platform Settings

Primary to the Platform UI is the Platform Settings page. This page supports administration of the Swarm platform, so that you can:

- View the current Platform configuration

- Reserve one or more sets of IP ranges



Platform Settings

Platform Version master

Platform IP 172.30.25.10

Platform OS Ubuntu Xenial

DHCP Subnet 172.30.25.0/24, Start 172.30.25.100, End 172.30.25.254

Reserved Range

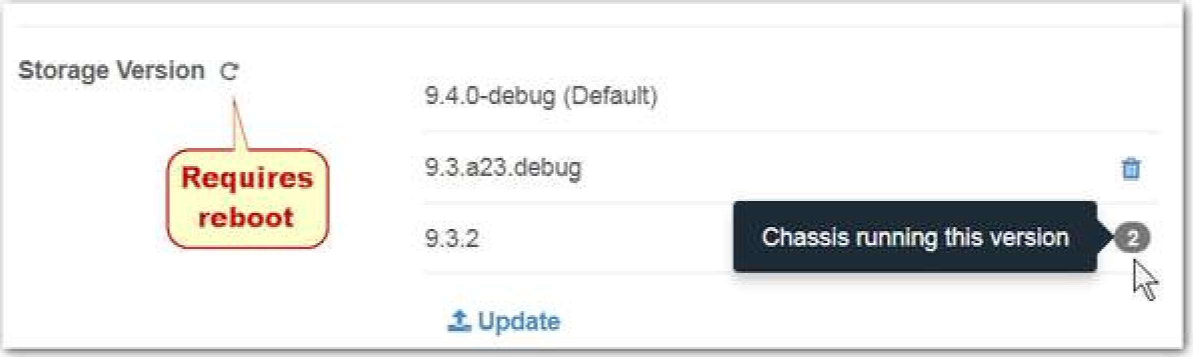
Start 172.30.25.50
End 172.30.25.60

Start 172.30.25.1
End 172.30.25.15

> Note
> You can safely add or modify a Reserved Range on a live cluster: if the range you set is invalid, Platform server will reject it.

Storage Version — The Platform Settings page also lets you manage your versions of Swarm Storage. You can:

- Upload a new version of Swarm Storage software (the ZIP file of the storage firmware boot image)
- See all versions that have been installed
- Delete an unneeded version, but not while any chassis is still using it (the chassis must first be changed to a different version and restarted)
- See how many chassis are currently running each version (a delete icon instead of a counter means there are none)



Storage Version

9.4.0-debug (Default)

9.3.a23.debug

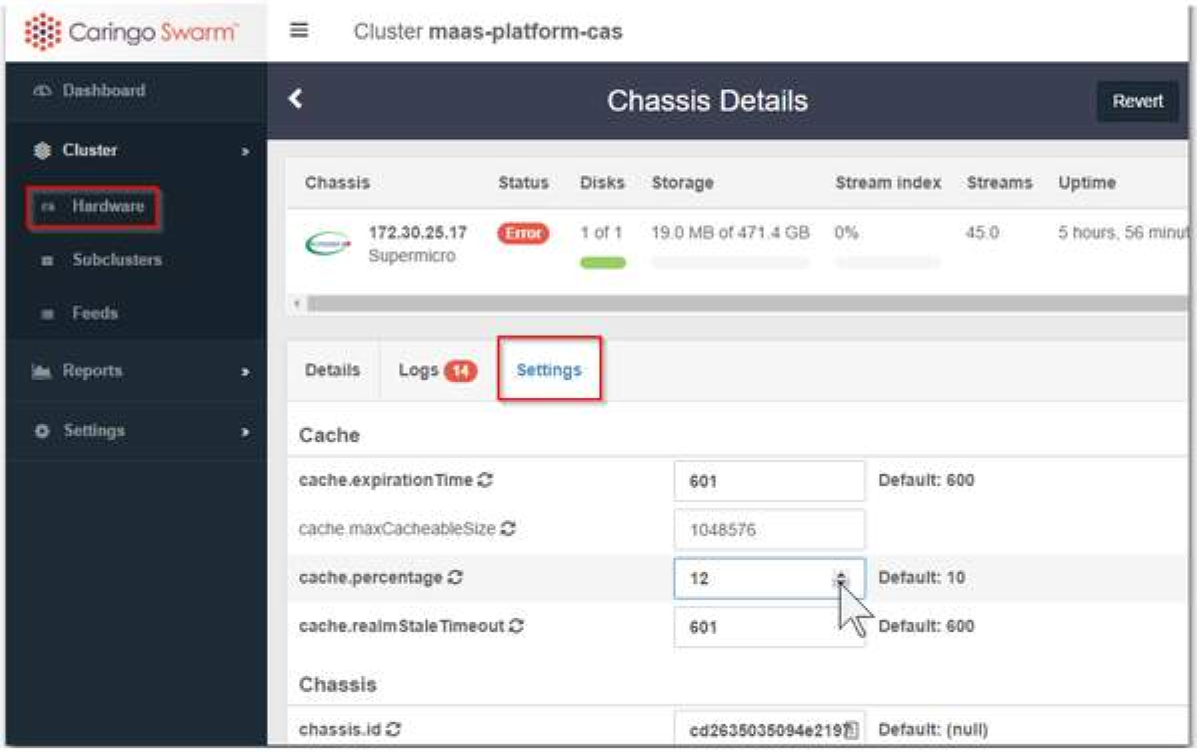
9.3.2

Chassis Settings

The other major area of Platform functionality is the Chassis Details, Settings tab, which shows all chassis-specific Swarm settings. There are two main reasons to edit these settings:

- Subclusters — Assign this chassis to a subcluster (`node.subcluster`)
- Archive — Put this chassis into or out of Archive mode (`node.archiveMode`)

> Caution
> Many settings available here are for expert usage. Get guidance from Support before changing other values on a particular chassis, such as ones controlling networking or Storage behavior.



Chassis	Status	Disks	Storage	Stream index	Streams	Uptime
172.30.25.17 Supermicro	Error	1 of 1	19.0 MB of 471.4 GB	0%	45.0	5 hours, 56 minut

Cache

cache.expirationTime	601	Default: 600
cache.maxCacheableSize	1048576	
cache.percentage	12	Default: 10
cache.realmStaleTimeout	601	Default: 600

Chassis

chassis.id	cd2635035094e219	Default: (null)

> Settings tips
> - Settings in bold highlight values that deviate from the Swarm default.
> - Settings with an icon next to them require a restart to take effect.

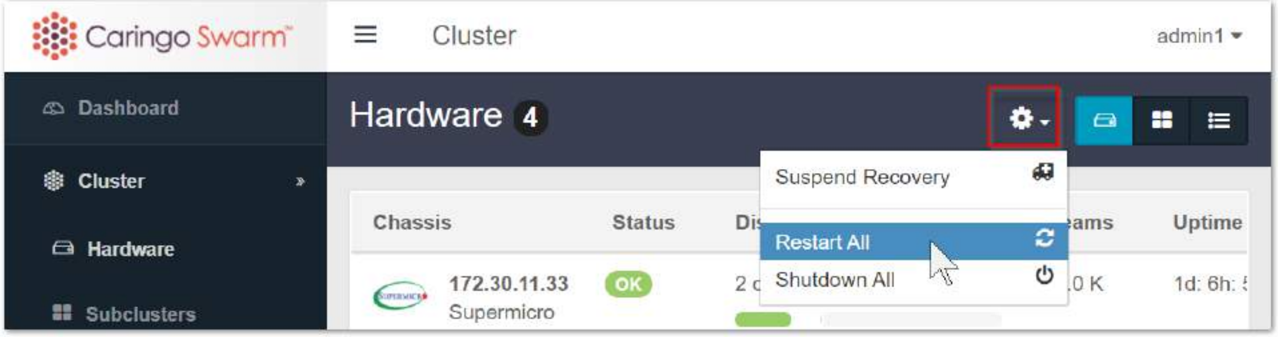
For information about a specific Swarm setting, hover over the setting name for a popup description.

> See the Swarm Storage Settings Reference.

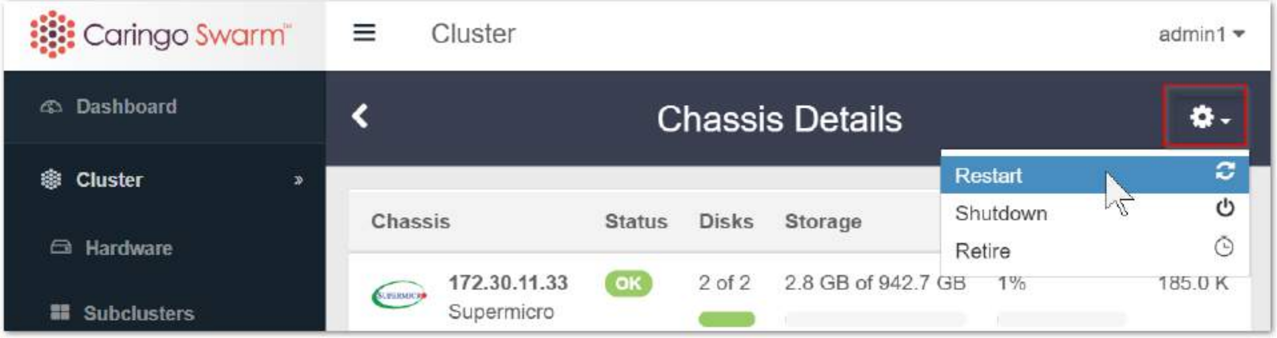
Restarting Hardware

From the Swarm UI, you can perform both a cluster restart (Restart All) and a chassis restart, so be sure to choose appropriately:

- From the Cluster > Hardware page, click the gear icon and select Restart All:



- From the Cluster > Hardware page, select a chassis; on Chassis Details, click the gear icon and select Restart:



After the chassis comes back online, proceed to the next chassis.

Platform Troubleshooting

- Adding Cores for Performance
- Listing Node State
- Listing Configured Settings
- Gathering Logs for Support
- Adding a Static IP Range
- Using Systemd
- Bootstrap Command Hangs
- Journalctl Issues

> Note
> Several of the following commands pull useful information through the REST API. The Platform Server has jq al ready installed to support these commands.

Adding Cores for Performance

> Swarm 10
> Swarm Storage 10 has a single-process architecture, so the configuration setting `chassis.processes` is no longer used and cannot be increased.

If you need to improve Swarm 9 performance under load, you can add additional processes (cores) beyond the default, which is 2.

1. In your hardware or in the VM settings, increase the number of CPU/cores.
2. In the Swarm UI or via CLI, configure the number of processes you want for each chassis:

   ```
   platform add config --name "chassis.processes" --value 3
   ```

3. Reboot all affected chassis.

Listing Node State

The following command queries the Platform Server for the list of nodes along with their state and other helpful information:

```
platform list nodes
```

Listing Configured Settings

Swarm settings can be changed through the Swarm UI or by uploading a new configuration (cluster.cfg) file. The Swarm UI uses bolding to indicate which configuration settings have been changed from the defaults.

To generate listings of only those setting values that have been explicitly configured in your cluster (versus every active default setting value), use this command:

```
platform list config
```

To list all settings (configured and default), use this command

```
platform list config --all
```

Gathering Logs for Support

Platform Server software comes with a utility to gather and tar up all of the logs and configuration information needed by Support. In order to run the script, you need to be a member of the following groups:

- adm
- syslog
- systemd-journal

To generate the file to send to Support:

1. Run the script:

   ```
   gather_logs.sh
   ```

   If you are not a member of the above groups and cannot become a member, you can run the script as sudo:

   ```
   sudo gather_logs.sh
   ```

2. When the script runs, it reports success with the location of the result:

   ```
   gather_logs.sh
   tar: Removing leading `/' from member names
   tar: Removing leading `/' from member names
   tar: Removing leading `/' from member names
   Logs have been gathered: ./logs-2018-04-02-11-29-52.tar.gz
   ```

3. Upload the log bundle on the Support site. The filename should have this form:

   ```
   logs-<year>-<month>-<day>-<hour>-<min>-<second>.tar.gz
   ```

Adding a Static IP Range

You cannot add a new static range that overlaps with the existing IPs; attempting this throws the following error:

```
platform add iprange -a 172.20.0.201 -z 172.20.0.250
Error: Could not add IP range: Invalid range, IP addresses already
allocated.
```

If you need to assign a new static range after deployment, it is possible to do so by temporarily releasing each chassis, adding the range, and then redeploying the chassis back into the cluster.

> Important
> To ensure a clean shut down, power off the chassis through the UI or SNMP before you run `release` commands.

1. List the deployed chassis and note the System ID:

```
platform list nodes deployed
```

2. One chassis at a time, release the chassis, add the new range, and redeploy it:

```
platform release storagechassis -y <system-id>
...
platform add iprange -a 172.20.0.201 -z 172.20.0.250
...
platform deploy storage -v <Swarm-version> -y <system-id>
```

3. Restart the cluster.

```
platform restart storagecluster --full
...or
platform restart storagecluster --rolling [<options>]
```

4. If using a Service Proxy, restart it:

```
platform restart proxy
```

Using Systemd

All of the containers and components on the Platform Server can be controlled using systemd.
To see the status of the four custom components, use these commands:

```
sudo systemctl zinc
sudo systemctl zincapi
sudo systemctl caringosimplestream
sudo systemctl etcd
```

To see how the containers are running with Docker, use this command:

```
sudo docker ps -a
```

To see journal information for the custom components, use these commands:

```
sudo journalctl -u zinc
sudo journalctl -u zincapi
sudo journalctl -u caringosimplestream
sudo journalctl -u etcd
```

Bootstrap Command Hangs

In the event the bootstrap command waits for the import process to finish and never returns, you may have entered the wrong IP address for the `reinit.sh` script. Use the following command to see the status of the platform server:

```
platform status platform
```

If the status shows "error", follow these steps to correct the problem:

1. Open `/etc/maas/rackd.conf` and find the `maas_url` setting. If the IP address matches the invalid address used for the reinit script, change it now (requires sudo access to edit)
2. Open `/etc/maas/regiond.conf` and find the `maas_url` setting. If the IP address matches the invalid address used for the reinit script, change it now (requires sudo access to edit)
3. Issue the following commands:

   ```
   sudo systemctl restart maas-regiond
   sudo systemctl restart maas-regiond
   ```

4. Use the `status` command and wait for the Status command to say `ready`.

   ```
   platform status platform
   ```

5. Reissue the bootstrap command.

Journalctl Issues

Following are useful tools for troubleshooting journalctl issues:

- iotop
- htop
- nmon
- bridge-utils
- sysdig (advanced)

Problem: The journalctl shows "`maas : problem with defaults entries ; TTY=unknown ; PWD=/ ;`"
Workaround: In `/etc/nsswitch.conf`, remove "`sss`" from the `sudoers` line. The change will activate immediately.
Reference: https://bugs.launchpad.net/ubuntu/+source/sssd/+bug/1249777

Swarm CSN Platform Server

> Separate Distribution
> This is a legacy Swarm component, with its own distribution packaging.

- CSN Overview
- CSN Requirements
- CSN Installation
- CSN Updates and Upgrades
- CSN Reset
- CSN Passwords
- Scenarios for Swarm UI
- Installing the Storage UI
- Migrating Elasticsearch
- Console - Cluster Services
- Console - Content Storage

CSN Overview

- CSN Console
- Swarm Storage UI
- Audience

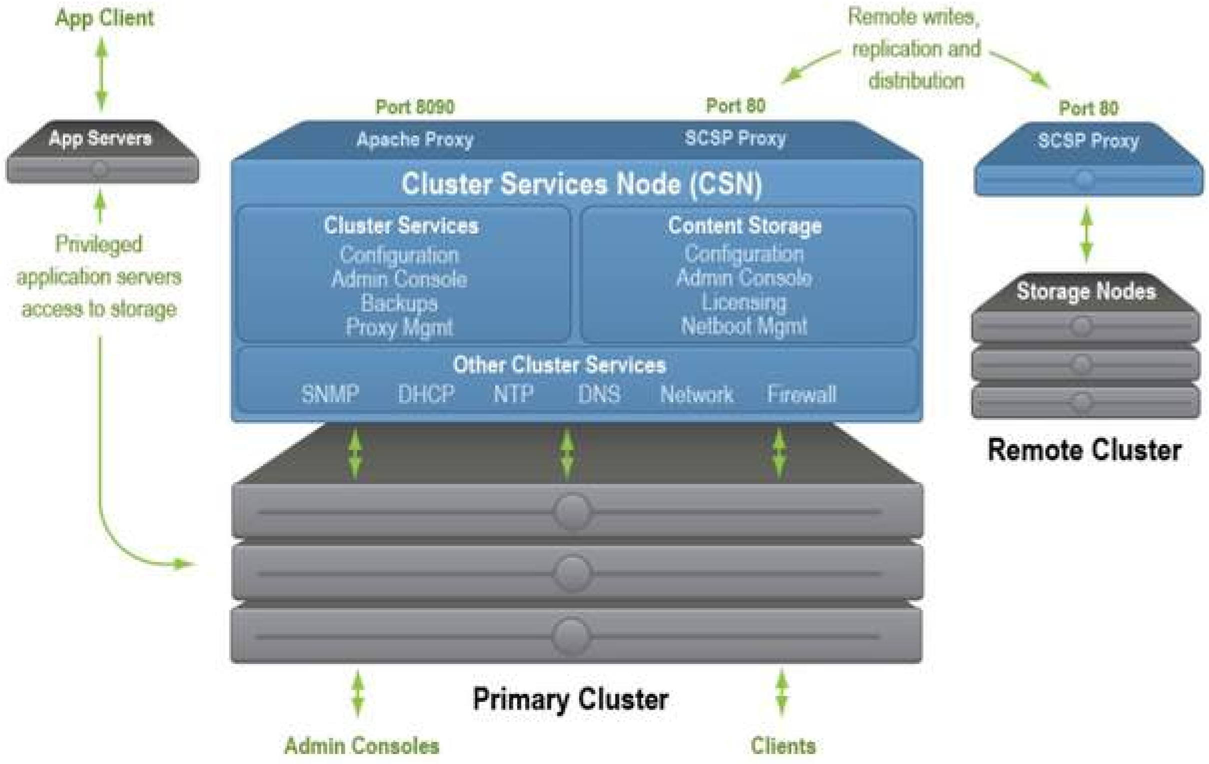
The Swarm CSN Platform Server (CSN) is an integrated services software application that centralizes the services and management information needed to install, upgrade, and monitor a Swarm storage cluster onto a single, accessible server with shared configuration. Installed on a dedicated node, CSN simplifies the network services and Caringo product installation process for administrators who are not proficient in Linux-based products or for environments where the network services are not easily available. Note that the CSN cannot anticipate all possible desired configurations. As a result, it cannot provide administrators the same flexibility as manually configuring the network services.

The CSN infrastructure is comprised of the following components:

- Required network services. These services are configured to support a Swarm cluster, which include DHCP, PXE/Network Boot, TFTP, Syslog, and NTP.
- Integrated PXE network boot and configuration server. For Swarm nodes booted onto the Swarm CSN's internal network.
- SNMP MIBs. Provides operational and status information for all nodes in the associated Swarm cluster.

- Swarm and SCSP Proxy. These integrated components are installed and configured for you.
- Administrative web console. Provides configuration of some settings and parameters, as well as access to useful utilities such as updating Swarm software versions and backup and restoring configuration files.

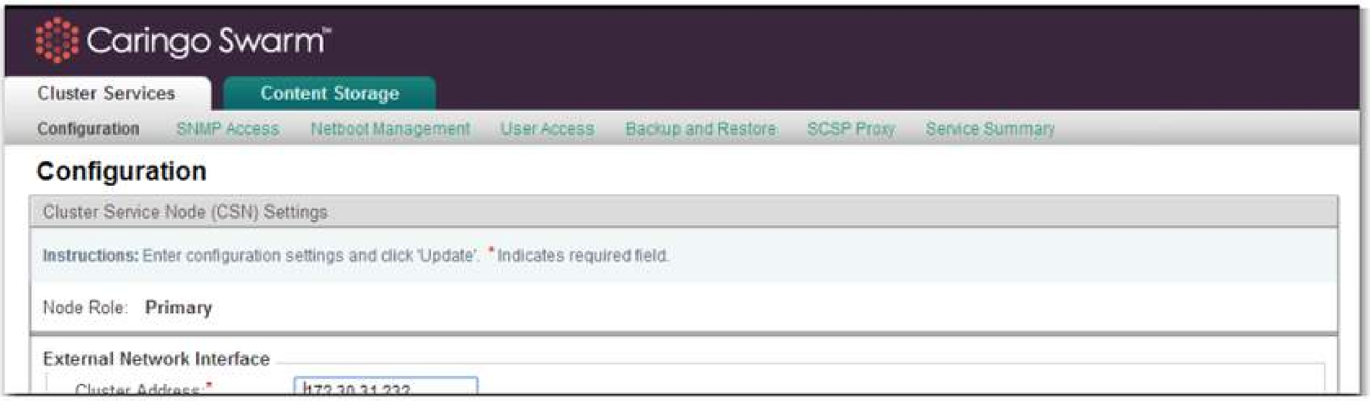
Below is a logical view of the CSN infrastructure. In a dual-network CSN, the primary storage cluster is isolated on a private network. On a single-network CSN, the primary storage cluster is directly addressable.



CSN Console

The CSN console is the primary administrative interface for the CSN, as well as other services and software installed on the shared server. Using the console you can manage basic system configuration changes and additional functions, such as creating manual configuration backups or changing Swarm software versions.

> Note
> JavaScript must be enabled in your browser when using the CSN console. Otherwise, the console will not function correctly and it will display a warning.



Configuration

Cluster Service Node (CSN) Settings

Instructions: Enter configuration settings and click 'Update'. * Indicates required field.

Node Role: Primary

External Network Interface

Cluster Address:* 172.30.31.232

- Cluster Services allows you to change CSN network parameters after initial configuration, SNMP and admin user access and SCSP Proxy configuration. You can also change the version of Swarm used for network booting storage nodes, create and view configuration backups and view a summary of all CSN services and their current status.
- Content Storage provides access to a Licensing interface and the ability to specify the MAC addresses for servers that should be formatted as Swarm nodes in the Netboot Protect interface.

> Warning
> Concurrent updates to the same CSN Console page from two or more browser sessions are not supported. Administrators should take care to ensure they are looking at a current version of the saved configuration prior to submitting any changes.

Swarm Storage UI

The Swarm Storage UI (website) presents a comprehensive browser interface for monitoring and controlling your entire Swarm storage implementation. It offers system and storage administrators a unified view of and easy access to the features and settings of Swarm:

- See all cluster chassis and disks, with both real time and historical status and metrics
- Initiate cluster and chassis-level actions like restarting or retiring disks
- View and change cluster settings
- Access event logs
- Identify disk volumes (using the drive light function)

> Accessing UIs
> The functionality of the legacy CSN Console and Admin Console have been unified and replaced by the Storage UI.

Site	CSN	Platform
Swarm UI	`http://<CSN·host>:<cluster_admin·bindPort>/_admin/storage`	`http://<Platform-serve`

Content UI	`http://<Gateway·IP>:<SCSP·bindPort>/_admin/portal`	`http://<Gateway·IP>:<S`
Legacy Admin Console	`http://<CSN·host>:8090/services/storage`	`http://<storage·node>`
Legacy CSN Console	`http://<CSN·host>:8090`	n/a

The `bindPort` refers to settings in the Gateway Configuration, and they offer the flexibility to support proxies and Docker environments.

Deprecated
The Legacy Admin Console (port 90) is still available, but it has been superseded by the Swarm UI. (v10.0)

> See Installing the Storage UI.

Audience

This guide is intended for storage system administrators, network administrators, and technical architects who understand TCP/IP networking and have a basic knowledge of setting up x86 systems in a corporate network. Additionally, the reader is expected to have a background in RHEL system administration, TCP/IP networking, and monitoring via SNMP. Throughout this document, administrator refers to Swarm CSN administrators who are normally responsible for software and hardware installation and configuration, license management, storage system health management, software upgrades, backup and recovery, and capacity management.

> See Storage Implementation, Swarm Storage Cluster, and SDK Overview.

CSN Requirements

- Hardware
- Operating System
- Networking
  - Dual-network
  - Single-network
- Pre-Installation Checklist
- About Primary vs. Secondary
- About monit

Hardware

The following table lists the recommended configuration for a CSN.

Parameter	Description
Node	x86 with 64-bit Intel® Xeon® or AMD® Opteron™ (or equivalent) CPUs
RAM	6 GB minimum, 12 GB recommended
OS	Red Hat® Enterprise Linux® (RHEL) 6 (release 6.8 or newer) Server 64-bit (English version) CentOS 6 (release 6.8 or newer) 64-bit Server Edition (English version)
Disk Storage	RAID configuration, for redundancy A separate partition with at least 16GB of available space configured for `/var/log`, to prevent logging from over-running other critical processes in `/var`.
Network	2 Gigabit Ethernet NICs minimum, 4 Gigabit Ethernet NICs recommended Single-network: Dedicated subnet that the CSN can control to allocate storage node IP addresses Dual-network: • Dedicated switch or VLAN for the internal storage network • Managed switch (recommended) • Either a disabled or properly configured switch spanning tree with a rapidly converging algorithm when using a secondary CSN

Operating System

Swarm CSN was developed and tested with the U.S. English versions of RHEL 6 Server 64-bit and CentOS 6 64-bit Server Edition with default repositories. Other versions/languages or Linux distributions are not currently supported. The CSN installation will fail if the RHEL/CentOS version is anything other than the English version of release 6.8+.

> Note
> When you install RHEL 6 Server 64-bit or CentOS 6 64-bit Server Edition, Caringo requires selecting the Basic Server installation type. Do not select Minimal, as this option may cause the CSN installation to fail.

Subsequent installation instructions will assume a pre-installed RHEL Linux environment with either internet connectivity to a default repository or an alternatively configured default yum repository for use in installing required third party packages. The CSN installation process is dependent on additional rpm packages from a default repository that are not installed on the system by default. Non-default repositories do not always contain the required version of all packages and are therefore not supported. Specifically, the tftp- server package required for network booting Swarm nodes must come from a default repository; the alternate atftp- server package is not supported.

If you are installing RHEL/CentOS from an ISO image, you must obtain the latest security fixes from a repository prior to installing or upgrading CSN. Specifically, for the GHOST security vulnerability you should update the glibc library (`yum -y update glibc`). If you install or upgrade RHEL/CentOS from an internet repository, the latest glibc library should already be included.

> Upgrade warning
> Upgrading to RHEL/CentOS 6.8+ while the EPEL repo is enabled will upgrade monit to version 5.14.1, which

breaks the `firstcsnboot` script. Do not enable the EPEL repo until after the CSN has been installed.

Networking

The CSN provides two options for network configuration:

- Dual-network — Isolates the Swarm cluster on a dedicated network so that the external network is isolated from both the PXE network boot server (including DHCP) and cluster multicast traffic. The CSN itself with a dual-network configuration is 'dual-homed' with half its network interfaces cabled to the external network and the other half cabled to the internal network. The storage nodes the CSN manages are cabled to the internal network only. This internal private network ensures the Swarm cluster traffic is protected from unauthorized access. In this network configuration, the CSN owns network interface detection and configuration, and it assumes there will not be any external manipulation of networking files or services. This configuration is ideal for installations without a lot of networking expertise and for those where cluster isolation is desirable.
- Single-network — Makes the Swarm nodes directly addressable on the same network as the CSN without requiring the SCSP Proxy to proxy requests. Both the CSN and the storage nodes it manages are cabled to the same network with single-network. In single network configurations, the CSN assumes a network administrator familiar with the environment in which the CSN runs will configure and maintain the CSN's network interfaces as well as basic network configuration, like configuring a static IP identity for the CSN. A single-network configuration is ideal for installations that have some networking expertise and want the CSN and Swarm nodes' IP addresses to reside in a larger subnet range.

Both network options allow definition of what servers the CSN will network boot as Swarm nodes. Single-network enables netboot protection by default to prevent accidental format of servers not intended as storage nodes on a larger network. Dual-network does not require netboot protection by default, but it can be enabled.

Note
The CSN does not support migration from one type of network configuration to another without reinstalling the CSN.

The following table summarizes the pros and cons of each networking option:

Feature	Dual-network	Single-network
Isolate cluster multicast traffic on a separate internal network, using SCSP Proxy to proxy requests to and from the external network	X	
Allow storage nodes to be directly addressable on a larger network without a proxy or an additional router connected to the private network		X
Requires a dedicated switch or VLAN for internal storage network	X	
Requires a dedicated routable subnet that the CSN can control for allocation of storage node IP Addresses		X
Automate detection and configuration of network interfaces and bonding	X	

Allow manual configuration of CSN networking and bonding by an experienced Linux administrator		X
Enable netboot protection to only allow network boot of Swarm nodes for explicitly defined servers	X (After enabling in the CSN console)	X (By default)

The following sections discuss additional details about each type of configuration to aid you in deciding which is more appropriate for your environment.

Dual-network

A dual-network CSN requires access to both the external network as well as a dedicated internal network. Allocation of the address space in the internal network is broken down as follows, depending on the network size selected during initial configuration (small or large):

Network Size	CSN	External Applications	DHCP	Swarm Netboot
small (/24)	`x.y.z.0-16`	`x.y.z.17-32`	`x.y.z.33-83`	`x.y.z.84-254`
large (/16)	`x.y.0.0-254`	`x.y.1.0-254`	`x.y.2.0-254`	`x.y.3.0 - x.y.255.254`

- CSN range — provides IP Addresses for the various services on the Primary and Secondary CSNs.
- External Applications range — provides for third-party applications that need to run on the internal network to interface with the Swarm cluster. All IP addresses in the third-party range must be static. If you are using CFS or FileFly with Swarm, the IP Address should be assigned in the third-party range. Best practice is to keep a list of all IP Addresses that have been allocated to different applications in the third-party range to prevent IP collisions.
- DHCP range — provides an initial, temporary IP Address to Swarm nodes during their initial boot until permanent addresses can be assigned to each Swarm process by the CSN. Other applications that used the CSN's DHCP server on the internal network would reduce the number of Swarm nodes that could be booted at the same time, which is why a separate third-party range has been provided. For a small network, the maximum number of servers that can be simultaneously booted from the provided DHCP range is 50. Once booted, there are a total of 170 IP Addresses in the Netboot range for a small network so 50 servers could run roughly 3 multi-server Swarm processes per node. Swarm nodes that support higher multi-server process counts require additional IP Addresses in the Netboot range, decreasing the number of nodes that can be booted. In a large network, the maximum number of Swarm nodes that can be booted at one time is 254.
- Netboot range — provides the IP Addresses seen in the Admin Console for all Swarm processes.

Cabling Dual-network Interfaces

A dual-network CSN requires at least one network interface each for the Internal and External networks. Additional available NICs would ideally be split between the two networks, alternating each NIC port's cabled connection across all network cards for redundancy (External, Internal, External, Internal, etc). The CSN's initial configuration script will detect how all NICs are cabled based on whether or not the specified external gateway is reachable from each NIC. The configuration script will print what it detects to allow you to verify correct cabling; see "Dual-network Primary Configuration" for details. Once confirmed during initial configuration, the CSN will bond all NICs assigned to each interface into a single bonded interface with balance-alb bonding.

Single-network

A single network CSN requires access to a dedicated subnet. The subnet must be at least a Class C range with 256 IP addresses and may be as large as a full Class B network. Important: The CSN itself must be configured with a static IP Address, subnet mask and gateway prior to CSN installation. The CSN's IP Address must be within the x.y.z.1 - x.y.z.16 range for the configured subnet. The gateway configured for a single-network CSN will also be used for the Swarm nodes when they are configured. See the RHEL/CentOS documentation for complete instructions on configuring networking. As an example, the following demonstrates what the manually configured em3 interface with a static 172.20.20.11 IP Address, 255.255.255.0 subnet mask and 172.20.20.1 gateway would look like in /etc/sysconfig/network-scripts/ifcfg-em3:

```
# CSN interface
DEVICE=em3
IPADDR=172.20.20.11
NETMASK=255.255.255.0
GATEWAY=172.20.20.1
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
MTU=1500
NM_CONTROLLED=no
```

Be sure to restart the network after configuring the IP Address (/etc/init.d/network restart). Similar to a dual-network configuration, a single-network CSN will divide up the allocated address space into ranges for the CSN, DHCP, and Netboot. The allocated ranges will vary depending on the size of the dedicated subnet and will be printed for confirmation during single-network installation.

Cabling Single-network Interfaces

All network interfaces on a single network CSN should be cabled to the same broadcast domain (VLAN) on one or more network switches. A minimum of two cabled network interfaces is still recommended for redundancy. In addition to configuring the CSN's static IP address, any desired bonding configuration (for either redundancy or throughput) should be completed by the administrator prior to starting single-network CSN installation. See the RHEL/CentOS documentation for complete instructions on configuring network bonding.

Pre-Installation Checklist

Before you install CSN, confirm all of the following:

- You have valid IP addresses for your DNS and NTP servers.
- For dual-network configurations: You must have two available external IP addresses: one to access the Primary CSN and another to access the CSN "cluster" (that is, the Primary CSN (required) and Secondary CSN (optional)). If you set up a Primary and Secondary CSN, you access the CSN console using the CSN cluster IP address in the event the Primary CSN is not available. For more information, see "Primary vs. Secondary" .

Before you continue, make sure these IP addresses are not already in use.

- For single-network configurations: You must have statically configured the IP address for the CSN as described in previous sections.
- All network interface adapters are connected properly. At least one network interface must be enabled during RHEL/CentOS installation so that it is available for use during CSN installation.

About Primary vs. Secondary

With two CSNs on an internal network, one CSN must be designated as the primary. A primary CSN is responsible for responding to DHCP requests and also for listening to all communication on the well-known IP addresses for the CSN's network(s). A secondary CSN can provide redundancy for all services on the primary CSN in the event of a disaster and can also provide scalability for incoming SCSP requests to the Swarm SCSP Proxy in dual-network configurations. To do this, you can either:

- Use the Swarm SDK to instantiate an SCSPClient that "load balances" across two or more Swarm SCSP Proxy(s).
- Use standard web load-balancing software or hardware in front of the SCSP Proxy(s).

> Important
>
> In the event of a failover from a Primary CSN to a Secondary CSN, the Secondary takes on the complete network identify of the Primary, which is why the Primary and Secondary CSNs must be installed on the same client VLAN. If the networks are completely separate, the Secondary will not be able to restore the Primary's configuration, as the restored IP Addresses will not be reachable on the Secondary's network.

About monit

The CSN installation script installs the watchdog process monit, which performs a variety of functions, including starting and restarting services if they fail unexpectedly. If monit is not already installed on your system, CSN installs and configures it automatically. When monit is installed, the following is added to your existing configuration:

- Changes `/etc/monit.conf` by adding the following:

```
#START Caringo additions
set daemon 15                     # check services at 15 second
intervals
set httpd port 2812 and
    use address localhost         # only accept connection from
localhost
    allow localhost               # allow localhost to connect to
server
set logfile /var/log/monit.log # logs to local file
##END Caringo Additions
```

- Installs several product-specific configuration files under `/etc/monit.d`. The file name ends with `.monitrc`
- Adds a product-specific file under `/etc/init/` to enable the upstart service to monitor monit so it runs when the machine restarts or if monit itself fails.

> Note
>
> Manual editing of monit configuration files is not supported.

This section describes how to install and configure your Swarm CSN on a dedicated node.

- Installing the CSN
- CSN Network Setup
- Swarm Licensing Setup
- Booting Swarm Nodes

Installing the CSN

The CSN distribution is available as a collection of RPM packages that are installed with a shell script. The packages and their dependencies must be installed as a user with root privileges from an attached monitor and keyboard with at least one NIC connected to the external network.

> **Requirements**
> You must log in to the CSN machine to install the software. You cannot install the CSN using an SSH session b ecause the network configuration can terminate the connection prior to completion. Do not use "sudo" or "su" for installing CSN as this can interfere with installation. (CSN-2066)

> **RHEL edition**
> During installation, administrators installing from the RHEL gnome desktop may see errors stating "An internal system error has occurred". This error is related to package update software running before the CSN is fully initialized and can be safely ignored. The Basic Server edition of RHEL is recommended.

> **Root passwords**
> CSN root passwords must be alphanumeric with no spaces or special characters.

1. As a user with root privileges, copy `caringo-csn-{version}.zip` to your Cluster Services Node and unzip it into an empty directory of your choice. This directory is referred to as csn-extract-dir. For example, if csn-extract-dir is /tmp/csninstall:

   ```
   cd /tmp/csninstall unzip caringo-csn-{version}.zip
   ```

2. Install the CSN by running the self-extracting script from the directory location where the shell script was unzipped. You must run the following command as a user with root privileges:

   ```
   cd caringo-csn-{version} ./caringo-csn-bundle-install.sh
   ```

   This command initiates installation of the CSN and its dependent packages. If you are upgrading an existing CSN, you will be prompted to reboot when the install process completes; no further configuration is required.
3. When a new installation is complete, the following prompt displays: Would you like to proceed with CSN network

configuration? (yes/no):

- yes - Proceed with configuring the CSN and continue to "Dual-network Primary Configuration" if you are installing the CSN in a dual-network configuration or "Single-Network Primary CSN Configuration" if you are installing the CSN in a single-network configuration within a larger network.
- no - Set up the network at a later time by running the command: `/opt/caringo/csn/bin/firstcsnboot`.

> Important
> - Before running `firstcsnboot`, review the "Pre-Installation Checklist".
> - Only use `firstcsnboot` to configure the network for the first time. To reinstall because of a failed installation or mistakes made during installation, run `csn-reset`.
> - Only run `firstcsnboot` from the local terminal console. It cannot be run over an `ssh` connection.

CSN Network Setup

After running the CSN installation script and answering yes to the question about CSN network configuration, you must enter some minimal configuration data to configure the server on the overall network. Network settings are central to all CSN services and should be planned with care in advance by an administrator knowledgeable about the environment. The initial configuration process is required only once after the initial installation. Any necessary subsequent updates to the initial configuration parameters can be made from the CSN Console. Several prompts will suggest a default value in brackets that you can accept by pressing Enter.

Dual-network Primary Configuration

The following prompts will display if you are configuring the Primary CSN on a dual-network:

Prompt	Description
Is this the Primary CSN (yes/no)? [yes]:	This prompt enables you to specify whether or not the CSN is a primary or secondary CSN. For a review of primary and secondary CSN, see "Primary vs. Secondary" . Make sure that only one primary CSN is configured on the internal network to prevent IP address conflicts as well as conflicts with both DHCP and Swarm netboot configuration. The primary server must be configured prior to configuration of a secondary. DHCP is not started on the secondary CSN.
Is this a single or dual network CSN (single/dual)? [dual]	Enter whether the CSN should be configured in a 'single' configuration with both the CSN and the storage nodes directly addressable on a single network or in a 'dual' configuration with the storage nodes isolated on an internal network. For a review of dual and single-network configurations, see "Networking" .
Half of the NIC ports on this system will be bonded and assigned to the external network. The following questions configure the external network:	The following message then displays if you are setting up a Primary CSN.

Enter the CSN IP address [].	Enter an available IP address on the public network. This is the address you use to access the CSN console and that external clients use to access resources on the internal network (such as the Swarm cluster through the SCSP Proxy).
Enter the cluster IP address. This IP address will remain with the Primary CSN in the event of a CSN failover []:	Enter another publicly accessible CSN IP address. (The term cluster here refers to a cluster of CSNs; that is, one primary CSN and one secondary CSN.) This well-known address remains with the primary CSN in the event of a failover, meaning the cluster can always be reached at this address. The entered address must be in IPv4 format and must not already be in use by another host on the network.
Enter the subnet mask [255.255.255.0]:	Enter the subnet mask for the preceding IP addresses. The default is 255.255.255.0.
Enter the gateway IP address []:	Enter the IP address of the default gateway for the preceding IP addresses.

<table>
<tr>
<td>Half of the NIC ports on this system will be bonded and assigned to the internal network. The following questions configure the internal network: Enter the network address, e.g. 192.168.100.0 (small network), 192.168.0.0, 172.20.0.0 (large network) []:</td>
<td>

Enter an IP address in your internal network to create an address space for Swarm nodes, third-party servers like DHCP, and other services. Enter the IP address in one of the following formats, depending on your needs:

- Large network (Class B - more than 128 nodes): Format like 192.168.0.0, 10.10.0.0, or 172.20.0.0
- Small network (Class C - 128 or fewer nodes): Format like 192.160.100.0, 10.10.100.0, or 172.20.100.0, where the 100 in the third octet limits the number of available IP Addresses in the specified rang

The interface is divided between the CSN(s), privileged applications on the internal network and the Swarm nodes. The initial configuration process automatically creates multiple alias IP addresses on the internal network for use by various system services and reserves similar IP Addresses for a Secondary CSN. Prompts similar to the following display:

```
Configuring external/internal ports. This may take some
time.
Checking ... eth0 ...eth1 ...eth2 ...eth3 ...
Eth Device | MAC | Public? | Bond
eth0 | 00:0c:29:e2:e6:65 | Y | bond1
eth1 | 00:0c:29:e2:e6:6f | Y | bond1
eth2 | 00:0c:29:e2:e6:79 | N | bond0
eth3 | 00:0c:29:e2:e6:83 | N | bond0
=============================================
Disconnected NICs =
Recommended ethernet device assignment
Internal NICs = eth2 eth3
External NICs = eth0 eth1
Input the list of External NICs.
```

The preceding shows how CSN suggests you allocate the available NICs by assigning them to internal and external CSN interfaces. The recommendation is based on how the NICs are currently cabled. All NICs that are currently connected to the network display. Any NICs that are not connected to the network display as Disconnected NICs. After you connect them to the network (or if you later need to change NIC assignments), you can assign them to internal or external CSN interfaces. See "Modifying CSN Network Configuration" .

Note: If you answer no to the last configuration prompt asking if the entered values are correct, the interface table displayed on subsequent runs through the configuration prompts will show the values you previously selected for external and internal NICs NOT newly detected values based on any cabling changes.

</td>
</tr>
</table>

external nics [space-separated-list]:	Displays the list of NICs the CSN has determined are for its external interface based upon responses from the external gateway. You can optionally change the list if you want to override the CSN's detection. If the NIC distribution is not as expected and you need to recable the NICs, best practice is to abort the initial configuration script, correct the cabling and then rerun initial configuration using /opt/caringo/csn/bin/firstcsnboot. Note: If the configured external gateway has ICMP turned off so that it does not respond to ping requests, the CSN will not be able to automatically detect how the NICs are cabled. You will need to manually assign the NICs to either the external or internal interface using the blank prompts.
internal nics [space-separated-list]:	Displays the list of NICs that CSN has determined are for its internal interface based on lack of response from the external gateway. You can optionally change the list if you want to override the CSN's detection. If the NIC distribution is not as expected and you need to recable the NICs, best practice is to abort the initial configuration script, correct the cabling and then rerun initial configuration using /opt/caringo/csn/bin/firstcsnboot.
Enter a list of IP addresses (separated by spaces) for external name servers [8.8.8.8 8.8.4.4]:	Enter a space-separated list of DNS server IP addresses for the external interface. The default values specify public DNS servers.
Enter a list of IP addresses or server names (separated by a space) for external time servers [0.pool.ntp.org 1.pool.ntp.org 2.pool.ntp.org]:	Enter a space-separated list of Network Time Protocol (NTP) server IP addresses or fully qualified host names for the external interface. The default values specify public DNS names from the NTP pool project .
Enter a unique storage cluster name. This name cannot be changed once assigned. A fully qualified domain name is recommended []:	Enter a unique name to identify the Swarm cluster. This value must be unique among all clusters you manage; otherwise, disaster recovery will be problematic. (If the contents of more than one cluster with the same name are written to a single disaster recovery cluster, it will be very difficult to recover the contents later.) Caringo strongly recommends you use a fully qualified IANA domain name (for example, csn.example.com). The name you enter displays as the name of the storage cluster on the CSN Console as well as in metadata for all objects written to the local cluster. The CSN also uses this name to detect all the nodes participating in the cluster.
Enter the multicast group that should be used to uniquely identify the storage cluster on the network. Different storage clusters on the same network must have unique multicast addresses or the nodes will merge into a single cluster (224.0.10.100):	Enter a multicast group address for the storage cluster. Each cluster on the same network must have a unique multicast group address. It must be a Class D IP address in the range 224.0.0.0 - 239.255.255.255.

Are these values correct (yes/no)?	This last step allows you to review the values entered for all prompts before submitting them. Entering yes enables the initial configuration process to proceed with network and service configuration, resulting in a fully functional CSN. Entering no to the final initial configuration prompt restarts the initial configuration script at the first prompt, with the previously entered values populated. Values other than yes or no (case sensitive) are not supported. Note: After you complete the installation, but before CSN reboots, Fail messages might display. These messages are harmless and can be ignored.

At the completion of a successful initial configuration, the CSN immediately reboots the server to initialize all services. When the node comes back up, all network services are configured and available, including SNMP v2, syslog, DHCP, DNS, NTP, and firewall. Additionally, the CSN Console is available and the SCSP Proxy will be configured and started.

Important

The CSN configuration of network services in a dual-network configuration assumes full system control. Manual editing of system network configuration files (for example, /etc/sysconfig/network) may have unexpected results, so it is not supported.

Dual-network Secondary CSN Configuration

After you configure a dual-network Primary CSN, you can optionally configure a Secondary CSN. The Secondary requires only the entry of a single unique external IP address and identification of the internal interface that is already defined on the Primary. The Secondary will then pull much of its network configuration data from the Primary. To ensure current configuration data is pulled, make sure a backup has occurred after the last configuration change on the Primary prior to installing the Secondary CSN. To facilitate this, a one-time use of the Primary's root password is required as follows (blank passwords are not supported):

```
Additional information about the network will be obtained from the primary CSN
Please enter the primary csn root password []:
Please re-enter the primary csn root password []:
```

Taken from the confirmation at the end of the initial configuration script, the Secondary configuration parameters are simply:

```
Primary: no
CSN Type: dual
External CSN IP address: 192.168.66.11
Internal network address: 172.20.20.0
external nics [defaults from system scan]:
internal nics [defaults from system scan]:
Are these values correct (yes/no)?
```

Note: If a server on the Secondary's public network has an address that conflicts with the private internal address (.3) of the Primary CSN, then discovery of the Primary will fail and the Secondary CSN will not be able to boot. Administrators should take care to ensure IP address spaces do not overlap.

Once configured, a secondary CSN will register with the Primary CSN on the internal network using a privileged SSH channel setup during configuration. This allows the primary to periodically sync needed information to the secondary, specifically the Primary's Backup Manifest UUID. Changes in the Primary's configuration are NOT automatically updated on the Secondary CSN, but will be restored from the Primary's last backup in the event of an admin-initiated Failover.

Single-Network Primary CSN Configuration

The primary CSN in a single-network configuration should already have its network interfaces and IP address configured prior to installation, so the configuration prompts do not include those settings in single-network. During the configuration process, the CSN will read the subnet that was statically pre-configured in the CSN's network services files and use it to define the ranges that will be used for DHCP and netboot. The ranges will be printed during the configuration process to verify they are as intended. Administrators only need to define that the configuration should be for a primary CSN on a single network and then provide name server IP Addresses(es), NTP server(s), and a unique name and multicast group to identify the Swarm cluster as described in the dual-network configuration above.

Taken from the confirmation at the end of the initial configuration script, the single-network primary csn configuration parameters are simply:

```
DHCP Range: 172.20.20.33 - 172.20.20.83
Netboot Range: 172.20.20.84 - 172.20.20.254
Primary: yes
CSN Type: single
Name Servers: 8.8.8.8 8.8.4.4
Time Servers: 0.pool.ntp.org 1.pool.ntp.org 2.pool.ntp.org
Storage cluster name: cluster.example.com
Multicast Group: 224.0.10.100
Are these values correct (yes/no)?
```

Note the DHCP and Netboot Ranges are not entered during configuration but derived from the configured subnet.

Single-Network Secondary CSN Configuration

The secondary CSN in a single-network configuration should also already have its network interfaces and IP address configured prior to installation. Similar to a secondary CSN in a dual-network configuration, the secondary in a single-network configuration gets most of the rest of its configuration information from the primary CSN. To ensure current configuration data is pulled, make sure a backup has occurred after the last configuration change on the Primary prior to installing the Secondary CSN. You will need to enter the primary's IP address and root password to allow the primary's configuration settings to be shared. Note that changes in the Primary's configuration are NOT automatically updated on the Secondary CSN, but will be restored from the Primary's last backup in the event of an admin-initiated Failover.

Taken from the confirmation at the end of the initial configuration script, the single-network secondary CSN configuration parameters are just these:

```
Primary CSN IP Address: 192.168.1.14
Primary: no
CSN Type: single
Are these values correct (yes/no)?
```

Swarm Licensing Setup

After you reboot post-configuration but before booting any storage cluster nodes, you need to publish the Swarm license file you received with your purchased capacity via the administrative CSN Console. The console allows web-based configuration of all CSN services after the initial network configuration.

To correctly publish a license, you need to upload the file:

1. Browse to the CSN Console: `http://{CSN·host}:8090`
2. Log in to the console. The username is '**`admin`**' and the default password is '**`caringo`**'.

3. Click the the Content Storage tab.
4. Click the Licensing link.
5. Next to Upload New License File, browse to the location of the license file you received with your published capacity.
6. Upload the license.

After the file has been updated, the licensing interface will show the company information populated in the license file as well as a last modified date for the license file.

The following characters are accepted: letters, numbers, and characters available by pressing Shift on the number keys (!, @, #, and so on).

> Note
> Swarm scans for license updates every 15 minutes; therefore, it may be 15 minutes before a newly published license is registered in a running storage cluster.

Booting Swarm Nodes

Prior to booting, you should make sure the CSN has had adequate time to sync with a reliable NTP source and is sync ready as a time source for the nodes. Enter the following at the command line:

```
$ ntpq -c rv
```

Check the returned output for these flags:

- 'sync_alarm' - indicates that the CSN is not yet ready to serve as an NTP server for the Swarm nodes on the internal network.
- 'sync_ntp' - indicates that all alarms have cleared; this can take as long as 20 minutes.

Once the CSN has been configured and rebooted, the NTP server is synced, and a Swarm license has been published with licensed capacity, you are ready to start booting Swarm nodes.

In a dual-network configuration, no additional steps are required; in a single-network CSN, you must first configure the nodes that should be recognized and netbooted.

CSN Updates and Upgrades

- Updating CSN 8.3, Storage, and Components
- Upgrading Swarm Storage Only
- Upgrading the CSN Version

To update any CSN implementation, download and use the CSN 8.3 bundle.

> Note
> Any components running separately from the CSN, such as Content Gateway and Elasticsearch, are not updated by the CSN install script and must be upgraded separately.

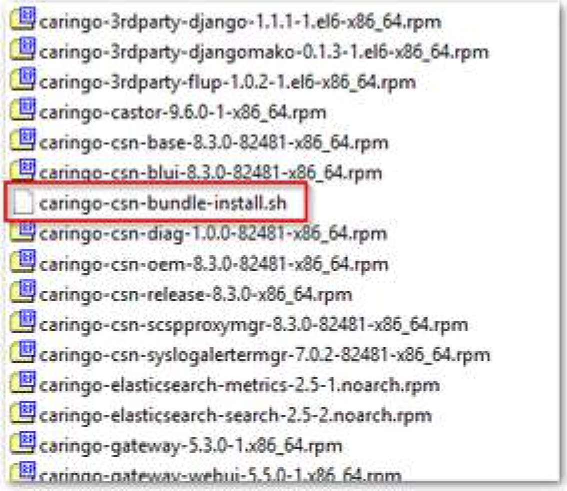
Updating CSN 8.3, Storage, and Components

If you are running the Swarm UI on the CSN itself, you update by running the CSN bundle installation script. To verify that you have Swarm UI on your CSN, log into it using this URL:

```
http://{CSN-IP}:91/_admin/storage
```

For this configuration, you can update the version of Swarm Storage software and all of the Swarm UI elements (Content Gateway, Swarm UI, etc.) by doing the following:

1. Download the latest CSN bundle.
2. Expand the bundle on your CSN server.

   ```
   caringo-3rdparty-django-1.1.1-1.el6-x86_64.rpm
   caringo-3rdparty-djangomako-0.1.3-1.el6-x86_64.rpm
   caringo-3rdparty-flup-1.0.2-1.el6-x86_64.rpm
   caringo-castor-9.6.0-1-x86_64.rpm
   caringo-csn-base-8.3.0-82481-x86_64.rpm
   caringo-csn-blui-8.3.0-82481-x86_64.rpm
   caringo-csn-bundle-install.sh
   caringo-csn-diag-1.0.0-82481-x86_64.rpm
   caringo-csn-oem-8.3.0-82481-x86_64.rpm
   caringo-csn-release-8.3.0-x86_64.rpm
   caringo-csn-scspproxymgr-8.3.0-82481-x86_64.rpm
   caringo-csn-syslogalertermgr-7.0.2-82481-x86_64.rpm
   caringo-elasticsearch-metrics-2.5-1.noarch.rpm
   caringo-elasticsearch-search-2.5-2.noarch.rpm
   caringo-gateway-5.3.0-1.x86_64.rpm
   caringo-gateway-webui-5.5.0-1.x86_64.rpm
   ```

3. Remove the yum lock before running the script:

   ```
   yum remove yum-plugin-versionlock
   ```

4. Run the CSN's install script from a terminal console, or run using SSH by adding the --viassh flag:

   ```
   ./caringo-csn-bundle-install.sh --viassh
   ```

5. From the Netboot page of the CSN Console, select the new Swarm version.

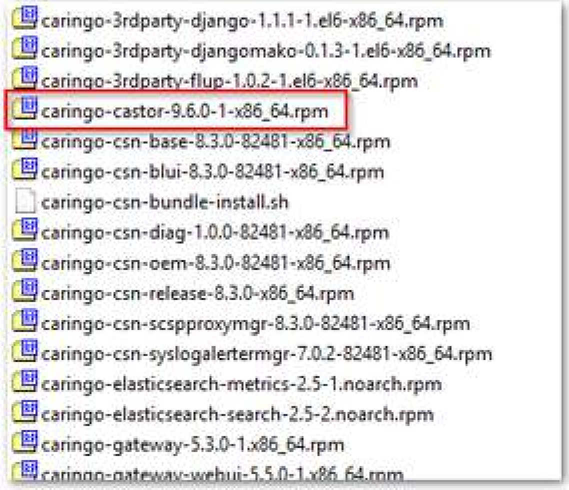
   > Important
   > This step is essential because the Swarm configuration values in the CSN may only be compatible with the newer Swarm software included in the CSN ZIP installation. Older Swarm versions may not boot with newer configuration parameters that are not recognized.

6. From the Swarm UI or legacy Admin Console, reboot the whole cluster at once or do a rolling reboot.

Upgrading Swarm Storage Only

To update the version of Swarm Storage software being used in the storage cluster, do the following:

1. Download the latest CSN bundle.
2. Expand the bundle on your CSN server.



caringo-3rdparty-django-1.1.1-1.el6-x86_64.rpm
caringo-3rdparty-djangomako-0.1.3-1.el6-x86_64.rpm
caringo-3rdparty-flup-1.0.2-1.el6-x86_64.rpm
caringo-castor-9.6.0-1-x86_64.rpm
caringo-csn-base-8.3.0-82481-x86_64.rpm
caringo-csn-blui-8.3.0-82481-x86_64.rpm
caringo-csn-bundle-install.sh
caringo-csn-diag-1.0.0-82481-x86_64.rpm
caringo-csn-oem-8.3.0-82481-x86_64.rpm
caringo-csn-release-8.3.0-x86_64.rpm
caringo-csn-scspproxymgr-8.3.0-82481-x86_64.rpm
caringo-csn-syslogalertermgr-7.0.2-82481-x86_64.rpm
caringo-elasticsearch-metrics-2.5-1.noarch.rpm
caringo-elasticsearch-search-2.5-2.noarch.rpm
caringo-gateway-5.3.0-1.x86_64.rpm
caringo-gateway-webui-5.5.0-1.x86_64.rpm

3. Install the new Storage (castor) RPM using the following command from a standard RHEL terminal window:

```
yum install [new Swarm version RPM]
```

4. After the install, open the CSN Console. A new Swarm software version will appear on the Cluster Services > Netboot Management page.
   See Console - Cluster Services.
5. To switch to an alternate Swarm software version, select the desired version and click Update. This will change the software image used to network boot all Swarm nodes.

> Note
> Suspending volume recoveries from the Swarm Admin Console is not necessary. Swarm will automatically enter into maintenance mode after an admin initiated reboot. Administrators may wish to delete older versions of Swarm software that are no longer needed to reduce the size of configuration backups.

6. Reboot the Swarm cluster so that it can pick up the new version. You may reboot the Swarm cluster all at once or rebooting one node at a time from the Admin Console to pick up the new version.

Upgrading the CSN Version

The CSN supports upgrade from one software version to the next via a script process similar to initial installation. The CSN will not be functional during the upgrade and will require a reboot after the software is updated. Schedule the upgrade during off peak hours to ensure Swarm activities are not disrupted.

CSN Migrations

If you are running CSN 2.x or are ready to upgrade to Platform 9.x, contact Support for help with your migration.

Backups

During upgrade, the backup process will not be able to complete and so may log errors. After upgrade, previous backup sets may be marked on the Backup and Restore page of the CSN Admin Console as being only compatible with a previous software version. These backups cannot be restored with the current version but will remain available if the software is reverted to a previous version. See CSN Reset for instructions on how to install different versions of the CSN using the csn-reset functionality.

You may upgrade CSN version 3.x or later:

1. Prepare OS - If you previously performed an OS version lock as required by older versions of CSN, remove the OS version lock using the following command:

   ```
   yum -y remove yum-plugin-versionlock
   ```

   If you are unsure if your OS was previously version locked, run the unlock command.
2. Upgrade OS - If the CSN's operating system is not running RHEL/CentOS 6 release 6.8+, you must upgrade the OS.
   - Similar to the initial installation, a yum repository must be configured and available for upgrade to proceed.
   - Either a media repository or an internet repository are acceptable. If you are installing from an internet repository, you may need to explicitly define the minor revision to which you want to upgrade. For example, from the Red Hat Network, you can select to receive Limited Updates Only and then chose the operating system version to be installed.
   - Depending on how long it has been since the server was last upgraded, the OS upgrade may take an hour or more.
3. Upgrade CSN - Upgrade the CSN software using the same install script and steps referenced in "Installing the CSN", with these notes:

   Important

   Before the CSN upgrade, if you have ever customized the `/etc/caringo/netboot/netboot.cfg` file (such as for "gateway" or "kernelOptions"), make a record of your non-default parameters.

   After the CSN upgrade, verify that these custom parameters were preserved; if not, update the config file and restart netboot: `service netboot restart`

   - Unless you have a reason to save them, you should agree to allowing the upgrade script to remove the Content Router data stores to reclaim the associated disk space.
   - New configuration parameters required to run Swarm 8.1 and later will be automatically converted during the upgrade process.

- You should also agree to the reboot option at the end of the installation; the CSN will not be functional until it has been rebooted.
- In the interim between upgrade and reboot, the Netboot component may log errors related to a port conflict. These errors are harmless and will be resolved with the reboot.

4. Upgrade Swarm - From the Netboot page of the CSN Console, select the new Swarm version.

> Important
>
> This step is essential because the Swarm configuration values in the CSN may only be compatible with the newer Swarm software included in the CSN ZIP installation. Older Swarm versions may not boot with newer configuration parameters that are not recognized.

5. Reboot - From the Swarm UI or legacy Admin Console, reboot the Swarm cluster.

After upgrade, no further configuration is required; all existing configuration and service states are preserved from their previously configured values.

CSN Reset

`csn-reset` enables you to reset a server to a state in which a fresh CSN software version can be installed from scratch and the initial configuration re-run. This can be useful to restart upgrades that are interrupted by network outages or to revert to a previous software version. Run the following command as a user with root privileges:

```
/opt/caringo/csn/bin/csn-reset
```

csn-reset will prompt for confirmation prior to proceeding and logs its steps to both the screen and to the system syslog (/var/log/messages). During the csn-reset process, configuration backups cannot complete and might log errors as a result.

> Note
>
> csn-reset is not a general-purpose uninstall script because it does not return service and network configuration files to their original state; it will not leave the server in a state usable for other purposes. To repurpose a server as a non-CSN, perform a fresh OS install.

For dual-network installations, if csn-reset fails with errors related to the public network not being reachable (for example, because of the way public bonds are configured), enter the following commands in the order shown as a user with root privileges:

1. `/opt/caringo/csn/bin/reconfigure-network`
   This command removes bonding from all network interfaces. See "Dual-network: Automatically Reconfiguring Network Settings".
2. `shutdown -r now`
   After reconfiguring the network, you must reboot.
3. /`opt/caringo/csn/bin/reconfigure-network --bonded --autodiscover`
   This command restores bonding to all network interfaces.
4. `shutdown -r now`
   After reconfiguring the network, you must reboot.
5. `/opt/caringo/csn/bin/csn-reset [--remove-cr]`

After having reconfigured the network, csn-reset should succeed.

CSN Passwords

With the CSN, you have passwords specific to the CSN in addition to the passwords for Swarm Storage.

- CSN-Specific Passwords
- Storage-Specific Passwords
  - Encrypting Passwords
  - Updating Passwords

CSN-Specific Passwords

These CSN-specific passwords can be managed from the CSN Console:

Password	Default User	Default Password	Update Location	Permission to:
CSN Console	`admin`	`caringo`	Cluster Services > User Access	Access the CSN Console (`http://{CSN` `st}:8090`).
SNMP Read-Only	n/a	`public`	Cluster Services > SNMP Access	Read (`snmpwalk, snmpget`) the SNMP MIBs available on the CSN.
SNMP Read-Write	n/a	`caringo`	Cluster Services > User Access	Read (`snmpwalk, snmpget`) and write (`pset`) to the aggregate storage MIB on t CSN.
Swarm UI	`caringoadmin`	`caringo`	From the root account's command line: `passwd caringoadmin`	Access the Swarm UI (`http://{CSN ho` `}:91/_admin/storage`) through the Service Gateway. The `caringoadmin` user is only for use b applications, not for logging in via ssh o the console (shell). Note To change this user, edit the root `dsys.json` and `policy.json` file on the CSN.

Storage-Specific Passwords

The Swarm Storage-specific passwords must be managed with these configuration methods:

Password	Default User	Default Password	Update Location	Permission to:

Default Swarm Administrator	`admin`	`caringo`	Via SNMP or Swarm UI	Modify settings and take actions such as Reboot on the Swarm UI or legacy Admin Console.
Additional Swarm Administrators	n/a	n/a		
SNMP Read-Write to Swarm nodes	n/a	`ourpwdofchoicehere`	/var/opt/caringo/netboot/content/cluster.cfg	Read (snmpwalk, snmpget) and write (snmpset) to the SNMP MIBs for the Swarm nodes directly.

The Swarm `admin` password is required to make changes in the Swarm Storage UI; read-only credentials (`operators`) are needed to view it. See Defining Administrators and Users.

> Important
> When you deploy your storage cluster, be sure to change the default passwords.

As of Swarm 10.0, there are two pairs of passwords to manage, which are in the cluster-wide `[security]` and `[snmp]` sections of the Swarm Storage configuration file (`cluster.cfg` (Platform/CSN) or else `node.cfg):`

Setting name	Default	Descriptions and Examples
security.administrators	`{'admin': 'ourpwdofchoicehere'}`	One or more username:password pairs. Sets credentials for who can administer the cluster via the Swarm UI. Upgrading from 9.x If the value includes the snmp username, remove it from here and update `snmp.rwCommunity` with its password. Example: `{'admin':'adminpassword','admin2':'adminpassword2'}`

security.operators	`{}`	One or more username:password pairs. Sets credentials for who can view the Swarm UI. Upgrading from 9.x If the value includes an snmp username, it is ignored; remove it from here and update `snmp.roCommunity` with its password. Example: `{'operator': 'operatorpassword','operator2': 'operatorpassword2'}`
snmp.rwCommunity	`ourpwdofchoicehere`	String. Password for the SNMP read-write community. Important You must know the SNMP read-write password in order to dynamically change the Swarm '`admin`' password. To change the SNMP read-write password, you must edit the config file.
snmp.roCommunity	`public`	String. Password for the SNMP read-only community.

Caution

- The name **`admin`** is reserved, so do not delete it, which could cause errors and affect performance. If you decide not to use **`admin`**, define a complex password to protect it.
- Swarm prevents cluster booting if the SNMP security administrator (read/write user) is not set properly in the configuration file.
- All administrative users and passwords must agree on all nodes or certain cluster actions will fail.
- Password updates are not complete until they are persisted in the cluster settings file across all nodes, and rapid, successive updates cannot be accepted on a given node until the first update completes processing.

Encrypting Passwords

Instead of a clear text password, you can represent the password as a hexadecimal-encoded MD5 hash of the following string:

```
username:user-list-name:password
```

where username and password consist only of ASCII characters and `user-list-name` can be either "CAStor administrator" or "CAStor operator".

To create the MD5 hash, use a programming language or a utility such as md5sum or Apache htdigest. For example, to update your node or cluster configuration file with a password hash you create using htdigest:

1. Create a file that contains a hash of the user name, password, and user list name:

```
htdigest -c password-file.txt "CAStor administrator" Jo.Jones
```

2. When prompted by htdigest, enter and confirm the user's password.
3. Open the new file (`password-file.txt`) in a text editor. The hash is the last entry in the string:

```
Jo.Jones:CAStor administrator:08b0468c1d957b7bac24463dd2191a2d
```

Updating Passwords

How you update the passwords depends on which ones need updating and whether Swarm has ever been started.

> Note
> If you do not use a Platform Server, make these changes to the `node.cfg`.

Situation	Process	Examples and notes

<table>
<tr>
<td>Swarm has never booted</td>
<td>1. Create and hash an admin password.
2. Update passwords in the config file.
3. Important: If booting from a USB flash drive, be sure to unmount/stop the USB drive or else the changes will not be saved.
4. Boot the Swarm cluster.
5. After the cluster is running, you can remove the password from the config file.</td>
<td>Hash of password
$ echo -n 'admin:CAStor administrator:NEWPASSWORD' | md5sum | cut -d ' ' -f1
7fe563b8532b3a460def0895895eebf5
The first time that you boot the cluster, the Swarm admin password must be in the config file:
[security]
administrators = {'admin':'7fe563b8532b3a460def0895895eebf5'}
When the cluster is running, Swarm stores the admin password in the persisted Settings object, at which point it is safe to remove the password from your configuration file for security purposes:
[security]
administrators = {}</td>
</tr>
<tr>
<td>Updating SNMP passwords</td>
<td>1. Update passwords in the config file.
2. Reboot the Swarm cluster.</td>
<td>Important
You must know the SNMP read-write password in order to dynamically change the Swarm 'admin' password. If you need to change the SNMP read-write password, you must edit the config file.
After rebooting with the new SNMP password in the file, you can proceed to change the Swarm 'admin' password.</td>
</tr>
</table>

Updating Swarm admin password	1. Create and hash an `admin` password.
	2. Update password via SNMP, which Swarm will save in the persisted Settings object.
		Changing admin password
		`snmpset -v2c -c SNMP- password -m +CARINGO-CASTOR-MIB Swarm- node- IP addModifyAdministrator s "admin:new-password"`
		`snmpset -v2c -c ourpwdofchoicehere -m +CARINGO-CASTOR-MIB 172.20.3.85 addModifyAdministrator s "admin:7fe563b8532b3a460def0895895eebf5"`

Frequently asked questions:

- How do I change the active SNMP read-write password? The SNMP passwords cannot be changed dynamically. Changing one or both requires a config file update and a cluster reboot.
- What is the SNMP read-only password? The read-only password '`public`', which is the 'community string'
- Is the read-only SNMP password in the persisted Settings object? No
- Can my SNMP read-write passwords in the persisted Settings object and cluster.cfg be different? Yes, but only the config file SNMP read-write password is used.
- How do I change my admin password? Update the password via SNMP and then update it in the config file, unless you've removed it from there.
- How do I change my SNMP read-only password to the cluster? Change the `snmp.roCommunity` setting in the config file and reboot the cluster.

Scenarios for Swarm UI

The installation scenarios below describe how to add Swarm UI to different CSN architectures. These are the components involved:

- Swarm UI is a web interface that connects via port 91 to Swarm nodes running Swarm 9.0+. Swarm UI must be installed on a server that has direct access to the Swarm cluster nodes.
- Service Proxy is a light (proxy-only) version of the Content Gateway that allows access to the Swarm UI web interface when the Swarm nodes are not open to the rest of the network (such as when the nodes are located in a dual-network CSN or a Content Gateway environment). You cannot use Service Proxy for writing content to the Swarm cluster: it is not for S3 or SCSP calls.
- Metrics Curator is a service that runs at set intervals to generate your Swarm historical metrics. The Swarm UI Dashboard and Reports sections display and graph these metrics.
- Elasticsearch stores the metrics data (indexes) but is not for Swarm Search. This cluster installs via the integration script, but it can be co-located on an existing node in your search cluster. A single-node ES cluster is enough to run the Metrics Curator, but without redundancy.

The integration script (`configure_storage_webui_with_serviceproxy.py`) included with CSN 8.3 installs and configures all of these required services.

Important

- Do not install the complete Content Gateway on a CSN. This configuration is unsupported.
- Do not install the Service Proxy on CSN versions prior to CSN 8.3.
- Only install the Service Proxy component of the Content Gateway using the included script, which also installs the Swarm UI.
- Ensure that your CSN has at least 6 GB RAM before adding these services.
- Ensure that you are running only one instance of the Metrics Curator across your entire Swarm implementation.

Network	Implemented	Unused	Installation Options
Dual	Storage cluster	Content Gateway Search cluster	Recommended: Run the integration script to install all of the required services for Swarm UI on your existing CSN.
Single	Storage cluster	Content Gateway Search cluster	Recommended: Run the integration script to install all of the required services for Swarm UI on your existing CSN.
Dual	Storage cluster Search cluster	Content Gateway	Recommended: Run the integration script to install all of the required services for Swarm UI on your existing CSN. Have the metrics use a new Elasticsearch instance on the CSN rather than the Swarm Search cluster you already have available. Alternative: Install Service Proxy, Swarm UI, and metrics curator on a new server and use the existing Search cluster for the Swarm UI's metrics collection.
Single	Storage cluster Search cluster	Content Gateway	Recommended: Run the integration script to install all of the required services for Swarm UI on your existing CSN. Have the metrics use a new Elasticsearch instance on the CSN rather than the Swarm Search cluster you already have available. Alternative: Install Service Proxy, Swarm UI, and metrics curator on a new server and use the existing Search cluster for the Swarm UI's metrics collection. Alternative: Install only the Swarm UI and the metrics curator on the CSN without installing the Service Proxy, pointing metrics collection to your existing Swarm Search cluster. Because all Swarm nodes have direct network access, the Service Proxy is not required. However, it is easier to install the entire package with the script provided and use the Service Proxy, which allows for flexible security (such as LDAP) to manage logins to the Swarm UI.

Dual	Storage cluster Content Gateway Search cluster		Recommended: Enable Service Proxy on one or more of the Content Gateways, as it is included in 5.1.2+. Then install Swarm UI on any webserver within the environment that has direct access to the nodes. Alternative: Install Swarm UI on one or more of the Content Gateways themselves. Metrics curator can also run on one of the Content Gateways or on the CSN (if available), but only one instance of the metrics curator can be running. In this environment, you can point the metrics at the existing Swarm Search cluster to avoid installing an additional Elasticsearch instance.

Installing the Storage UI

To use the Swarm Storage UI with Swarm Metrics, you need all of the enabling components, such as a simplified Gateway (Service Proxy) to provide the needed access and an Elasticsearch service and curator. Your CSN download includes all of the RPMs and scripts needed to perform the installation and configuration of these components.

Caution

- Do not using the CSN-embedded ES server as a Swarm Search Feed target.
- Do not change the preconfigured defaults for Metrics, which are set in `/etc/caringo-elasticsearch-metrics/metrics.cfg`
  - `metrics.target` and `metrics.port` are preconfigured for the Service Proxy.
  - Metrics are checked every 15 minutes and are retained for 120 days.
- If you have any scaling concerns, consult Support about whether you need a dedicated Elasticsearch cluster.

1. Install the CSN bundle as usual (the new RPMs will install for you).
2. Add a Swarm license and start up the Storage nodes.
3. Run the integration script to configure the Service Gateway, adding arguments as appropriate:

```
/usr/bin/configure_storage_webui_with_serviceproxy.py
```

Optional arguments	Default	Purpose
-h, --help		Display help summary and then exit.
-v, --verbose	INFO	Add to generate more extensive logging output.
-d DIR, --directory DIR	/var/log/caringo	Specify a different working directory for logs and output.
-s, --start_services	yes	Disable automatic enabling and start up of services after configuration.
-n NUM_HOSTS, --num_hosts NUM_HOSTS	5	Specify a different number of hosts to add to the Gateway configuration.

--indexer_host INDEXER_HOST [INDEXER_HOST ...]		Specify remote Elasticsearch hostnames or IPs, disabling (and preventing creation of) a local ES instance. Important: If you have an existing ES cluster, be sure to specify it here to avoid creating an extra local instance.

4. Each time the script prompts you for a configuration setting, press Enter to accept the suggested value, or enter your own.
5. On completion, the script will attempt to start all of the services with their new configurations, logging the output to the `integration.log` file (which defaults to `/var/log/caringo`).
6. When the script provides you with your URL and credentials, log into the Storage UI with a JavaScript-enabled browser (development and testing were conducted using the most recent versions of Firefox and Chrome).

> Note
>
> Metrics data will not appear until new Storage data is added and metrics collection cycles have occurred, which is a minimum of 30 minutes.

7. The script creates a "`caringoadmin`" user (password "`caringo`"), which is a user that is local to the CSN. These user credentials are required to login to the Storage UI through the Service Gateway.

> Note
>
> This `caringoadmin` user is only used by the Service Gateway, not by the Swarm cluster.

   a. To change the `caringoadmin` password on the local CSN, run the password command and follow the prompts:

```
passwd caringoadmin
```

   b. The user `caringoadmin` user is defined as the administrator for the Service Gateway in this file on the CSN: `/etc/caringo/cloudgateway/policy.json`. The user can be changed according to your requirements, just as in the Content Gateway.
   See Gateway Access Control Policies.

8. With CSN 8.3, all of these interfaces remain available and may be used concurrently:

Legacy CSN Console	`http://{CSN·host}:8090`
Legacy Admin Console	`http://{CSN·host}:8090/services/storage`
Swarm Storage UI	`http://{CSN·host}/_admin/storage`

Migrating Elasticsearch

The CSN installs a limited-size Elasticsearch instance to support Swarm Metrics. If you find that there is too much load on the CSN or that you would prefer to run a full ES cluster elsewhere, you can use the following method to migrate

Elasticsearch off of your CSN.

This technique lets you merge two separate elasticsearch clusters or rename an elasticsearch cluster, which is done by retiring an Elasticsearch node "into" a new Elasticsearch cluster.

1. Join one or more existing nodes to the new cluster by changing their `cluster.name` value to the new cluster's name.
2. Verify that they joined.
3. To migrate shards off of the old nodes, you decommission a node by telling the cluster to exclude it from allocation.

   ```
   curl -XPUT localhost:9200/_cluster/settings -d '{
     "transient" :{
         "cluster.routing.allocation.exclude._ip" : "10.0.0.1"
      }
   }';echo
   ```

   This causes Elasticsearch to allocate the shards on that node to the remaining nodes, which is done without the state of the cluster changing to yellow or red (even if you have replication 0).
4. When all of the shards are reallocated, you can shut down the node.
5. To restore the node to service, include the node for allocation, which causes Elasticsearch to rebalance the shards again.

> See the Elasticsearch documentation.

Console - Cluster Services

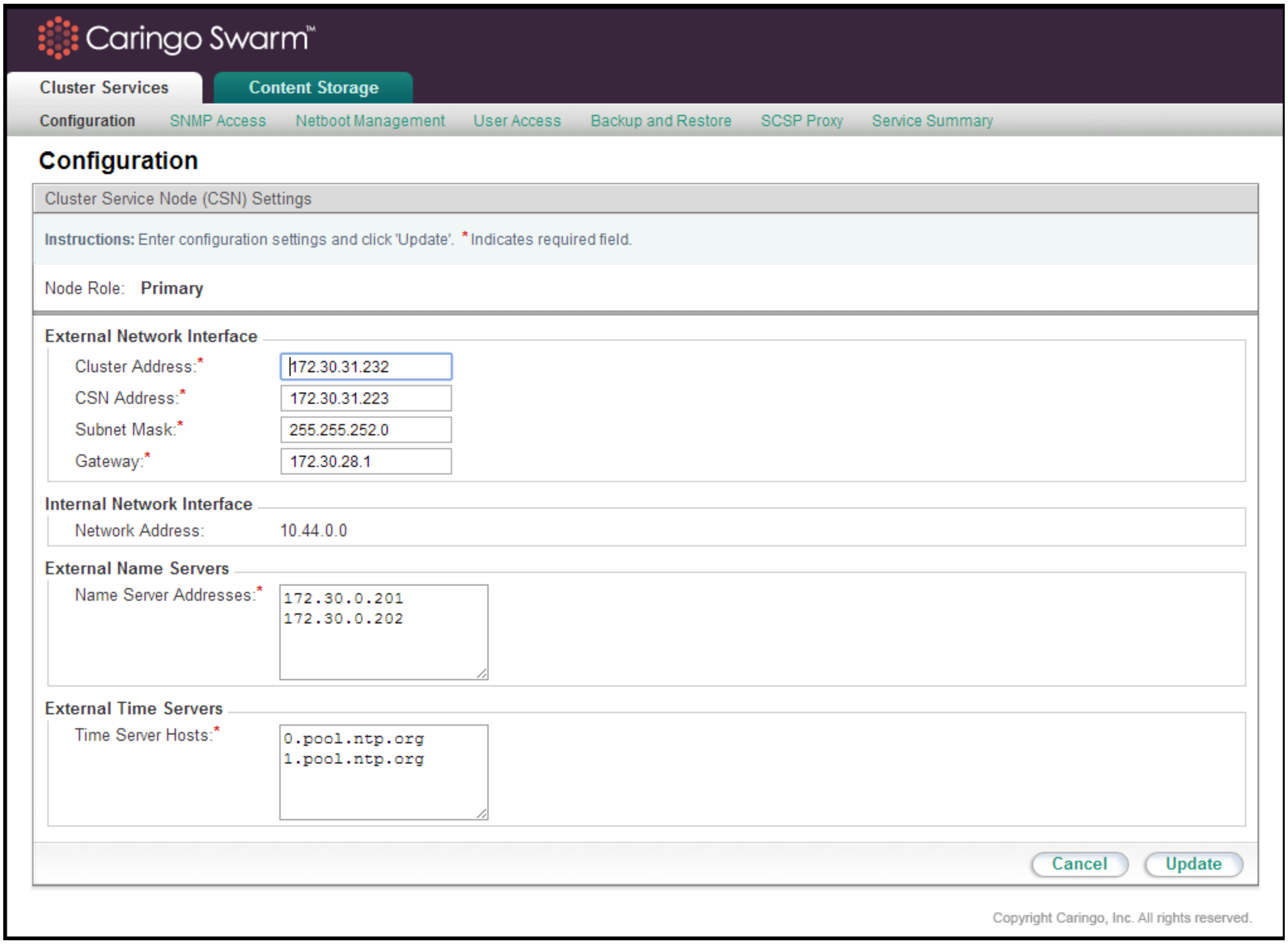
The CSN Console is the primary administrative interface for the CSN as well as the other services and software installed on the shared server. It allows both basic configuration changes as well as more complex functions like creating manual configuration backups or changing Swarm software versions.

- Configuration (network and servers)
  - Dual-network: Automatically Reconfiguring Network Settings
  - Dual-network: Updating the Internal Network
- SNMP Access (read-only commands)
- Netboot Management (Swarm version)
- User Access (CSN console)
- Backup and Restore (CSN configuration)
  - Backup Domain
  - Backup Manifest
  - Automatic Backups
  - Manual Backups
  - Restoring a Backup
  - Failover
- SCSP Proxy (dual-network configuration)
- Services Summary (versions and status)

- Syslog Configuration
- CSN Support Data Collection

Configuration (network and servers)

Administrators can update network parameters specified during the initial configuration process via the Cluster Services > Configuration interface.



Configuration

Cluster Service Node (CSN) Settings

Instructions: Enter configuration settings and click 'Update'. *Indicates required field.

Node Role: **Primary**

External Network Interface

Cluster Address:* 172.30.31.232
CSN Address:* 172.30.31.223
Subnet Mask:* 255.255.252.0
Gateway:* 172.30.28.1

Internal Network Interface

Network Address: 10.44.0.0

External Name Servers

Name Server Addresses:*
```
172.30.0.201
172.30.0.202
```

External Time Servers

Time Server Hosts:*
```
0.pool.ntp.org
1.pool.ntp.org
```

> Note
> In a single-network configuration, the network settings are read-only, because a single-network CSN does not configure the CSN's networking.

When changing values for any network parameter, the console may return a list of resulting actions that will need to be taken by each of the services and installed software modules. Because the network configuration is central to so many services, it is not unusual for the resulting action list to contain service restarts and possibly a server reboot.

Dual-network: Automatically Reconfiguring Network Settings

If the network in which the CSN is deployed changes, you can run a script to update the CSN's network bonding configuration. You can do this for any of the following reasons:

- After adding, removing, or replacing a NIC or after reconfiguring cabling, both of which can result in incorrect bonding configuration.
- After rebuilding the network from the current configuration in the event of misconfiguration (for example, after manually editing configuration files or if an external process modified the network configuration).
- As part of reinstalling the CSN because you need to reset the entire network configuration.

> Important
> These commands must be entered on the local console. You cannot execute them over ssh.

Command syntax follows:

```
/opt/caringo/csn/bin/reconfigure-network [-bonded [ -autodiscover]]
```

Argument	Description
none	Removes bonding from all network interfaces. Sets up an unbonded network on eth0 and removes any other ethx or bonded network configuration. To determine the network configuration for eth0, the utility first examines the current CSN configuration but, if none exists, it uses the public network bond defined by ifcfg-bond1.
`--bonded`	Sets up public and private bonds using the current CSN configuration. If that information does not exist or is not complete, the command fails.
`--bonded` `--autodiscover`	Automatically discovers all internal and external network adapter information and sets up public and private bonds using the current CSN configuration.

Reconfiguring All Network Settings

To completely reconfigure the network settings in a dual-network configuration:

1. Enter the following command as a user with root privileges: `/opt/caringo/csn/bin/reconfigure-network` This removes bonding from all network interfaces.
2. When prompted, reboot the machine.
3. After the server has rebooted, enter the following command as a user with root privileges:
   `/opt/caringo/csn/bin/reconfigure-network --bonded --autodiscover`
   Prompts similar to the following display:
   ```
   Configuring external/internal ports. This may take some time.
   Checking ... eth0 ...eth1 ...eth2 ...eth3 ...
   Eth Device | MAC | Public? | Bond
   eth0 | 00:0c:29:e2:e6:65 | Y | bond1
   eth1 | 00:0c:29:e2:e6:6f | N | bond0
   ```

```
eth2 | 00:0c:29:e2:e6:79 | N | bond0
eth3 | 00:0c:29:e2:e6:83 | Y | bond1
=============================================
Disconnected NICs =
Recommended ethernet device assignment
Internal NICs = eth1 eth2
External NICs = eth0 eth3
Input the list of External NICs.
```

The preceding displays how CSN has allocated your available NICs by assigning them to internal and external CSN interfaces. All NICs that are currently connected to the network display. Any NICs that are not connected to the network display as Disconnected NICs. After you connect them to the network, you can assign them to internal or external CSN interfaces by running this script again.

4. `external nics [space-separated-list]:`
   Displays the list of NICs that CSN has determined are for its external interface. You can optionally change the list if you want.
5. `internal nics [space-separated-list]:`
   Displays the list of NICs that CSN has determined are for its internal interface. You can optionally change the list if you want.
6. `Are these values correct (yes/no)?:`
   Enter yes to confirm the NIC assignments are correct or no to change them.

Messages display as changes are made and processes are restarted.

Restoring Current Network Settings

To rebuild the current configuration, enter the following command as a user with root privileges:

```
/opt/caringo/csn/bin/reconfigure-network --bonded
```

Messages display as the network interfaces are reconfigured. When prompted, reboot your machine.

Dual-network: Updating the Internal Network

Updating the internal network for a CSN can be extremely disruptive because it requires a coordinated update between the CSN and the Swarm cluster to ensure the two are using the same internal network. Updates to the internal network should be made only when absolutely necessary, using the following command:

```
/opt/caringo/csn/bin/changeinternalnetwork 172.20.0.0
```

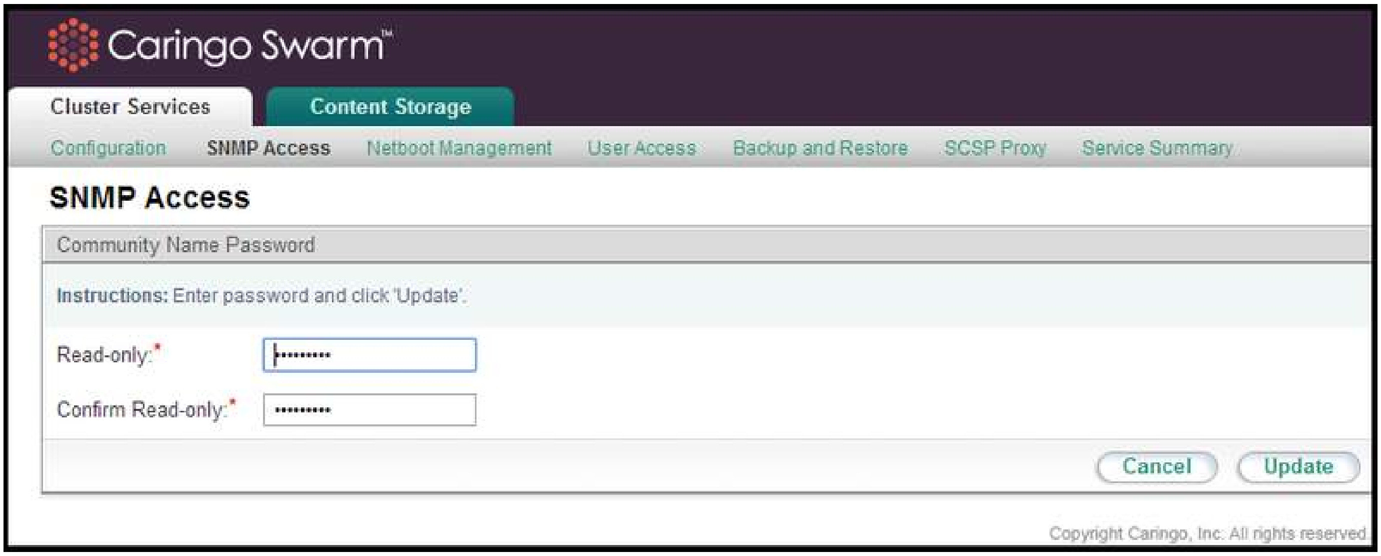
where 172.20.0.0 is the desired new internal network.

Best practice is to reboot the Swarm cluster from the Swarm Admin Console and then run the `changeinternalnetwork` script while the cluster is rebooting so that when the nodes come back on line they are assigned a new IP address from the new internal network interface range.

Alternatively, you can run the script and then manually power cycle the nodes; the Swarm Admin Console will no longer be able to route the reboot request after the internal network has been updated. Several CSN processes, including the NTP server, will be restarted when the script completes. After the Primary CSN has changed its internal network and the storage cluster is back on line, you should change the Secondary CSN's internal network to the same as the Primary.

SNMP Access (read-only commands)

The CSN installs the Net-SNMP command line tool for use in collecting SNMP data from the aggregated storage MIB, CARINGO-CASTOR-MGR-MIB. The tool utilizes standard SNMP commands like 'snmpwalk', and 'snmpget'. For read-only commands (snmpwalk, snmpget), the CSN expects a password of 'public' by default. The read-only default password may be updated from the Cluster Services > SNMP Access link of the CSN Console:



SNMP Access

Community Name Password

Instructions: Enter password and click 'Update'.

Read-only:* ••••••••••

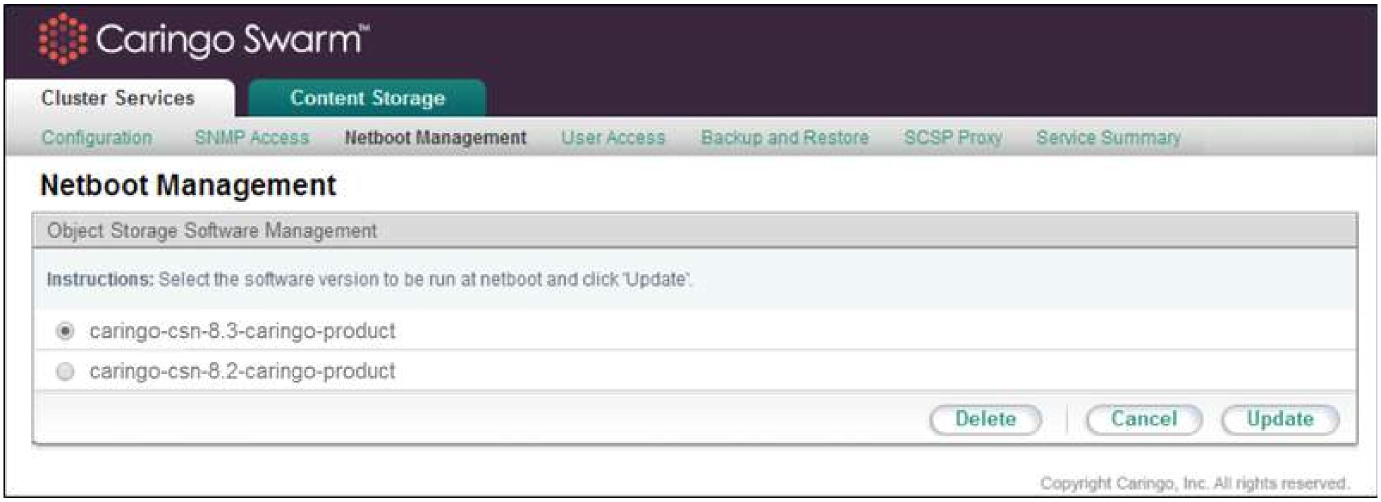
Confirm Read-only:* ••••••••••

Netboot Management (Swarm version)

To update the version of Swarm software being used in the storage cluster, you must install the new rpm using the following command from a standard RHEL terminal window:

```
yum install [new Swarm version]
```

After the install, a new Swarm software version will appear on the Cluster Services > Netboot Management page of the CSN Console:



Netboot Management

Object Storage Software Management

Instructions: Select the software version to be run at netboot and click 'Update'.

● caringo-csn-8.3-caringo-product

○ caringo-csn-8.2-caringo-product

To switch to an alternate Swarm software version, select the desired version and click 'Update'. This will change the software image used to network boot all Swarm nodes. The Swarm cluster must be rebooted to pick up the new version. You may reboot the Swarm cluster all at once or with a rolling upgrade one node at a time from the Swarm admin console to pick up the new version.

Note: Suspending volume recoveries from the Swarm admin console is not necessary. Swarm will automatically enter into maintenance mode after an admin initiated reboot. Administrators may wish to delete older versions of Swarm software that are no longer needed to reduce the size of configuration backups.

User Access (CSN console)

The user name for accessing the CSN console is '`admin`'. The admin password defaults to '`caringo`' and is updated from the Cluster Services > User Access link.

For Swarm 7.2 and later, you must change the Swarm Storage admin password via an SNMP command to the Swarm cluster (or the Swarm Storage UI). From the CSN, run a command like the following:

```
snmpset v2c -c <CURRENT- READ/WRITE- SNMP- PASSWORD> -m +CARINGO
CASTOR-MIB <SWARM- NODE- IP>
 addModifyAdministrator s "admin:<NEW- READ/WRITE- ADMIN- PASSWORD>"
```

Example:

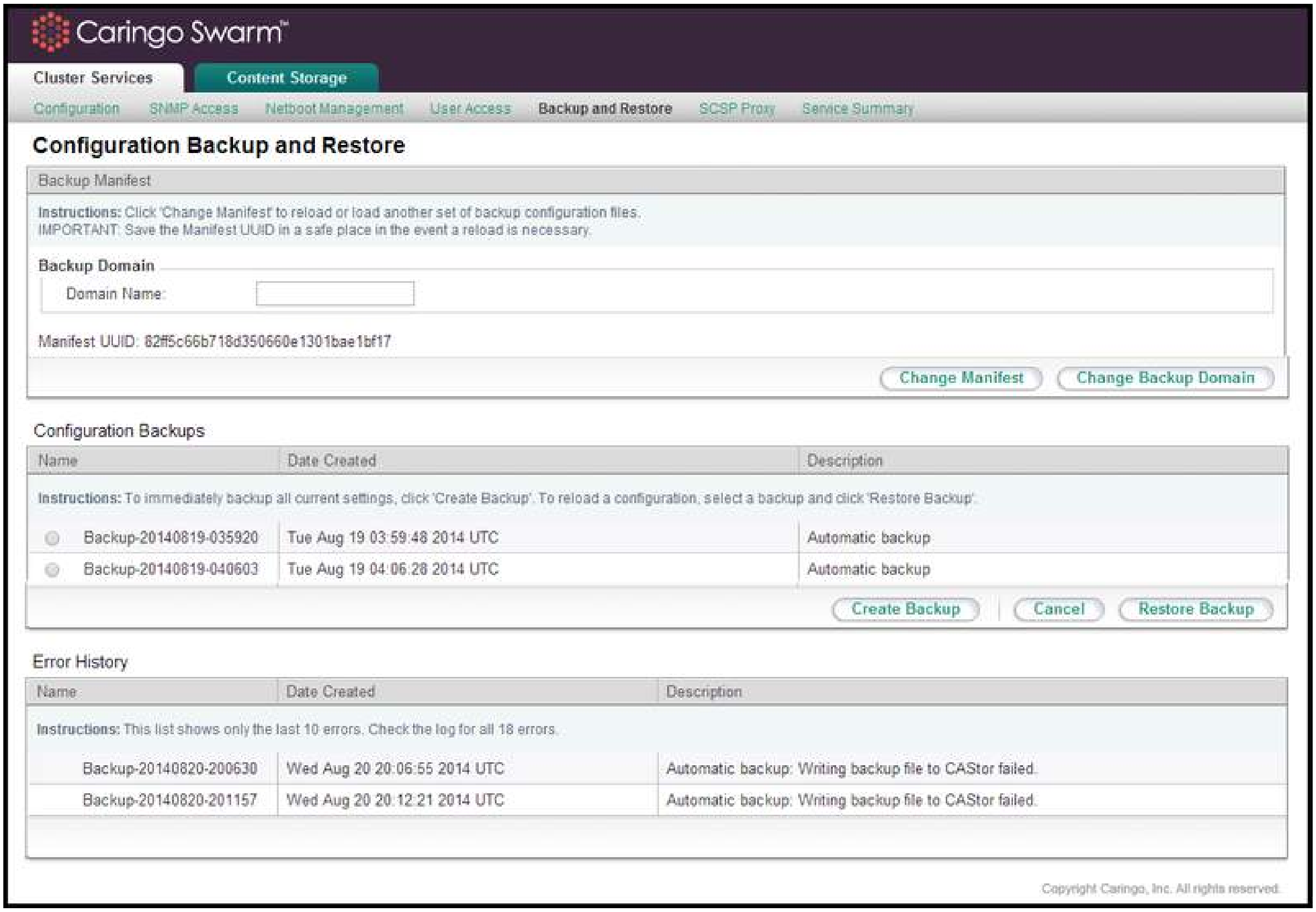
```
snmpset -v2c -c ourpwdofchoicehere -m +CARINGO-CASTOR-MIB 172.20.3.85
addModifyAdministrator s "admin:newpassword"
```

Tip

You do not need to reboot Swarm nodes if you have correctly updated the password via SNMP or the Swarm Management UI.

Backup and Restore (CSN configuration)

To help with recovery in the event of a hardware failure, the CSN includes a backup mechanism that backs up its configuration files to Swarm and then allows restore of those backups at a later date. The backup list and functions can be accessed from the Cluster Services > Backup and Restore link of the CSN Console:



Configuration Backup and Restore

Backup Manifest

Instructions: Click 'Change Manifest' to reload or load another set of backup configuration files.
IMPORTANT: Save the Manifest UUID in a safe place in the event a reload is necessary.

Backup Domain

Domain Name:

Manifest UUID: 82ff5c66b718d350660e1301bae1bf17

Configuration Backups

Instructions: To immediately backup all current settings, click 'Create Backup'. To reload a configuration, select a backup and click 'Restore Backup'.

Name	Date Created	Description
○ Backup-20140819-035920	Tue Aug 19 03:59:48 2014 UTC	Automatic backup
○ Backup-20140819-040603	Tue Aug 19 04:06:28 2014 UTC	Automatic backup

Error History

Instructions: This list shows only the last 10 errors. Check the log for all 18 errors.

Name	Date Created	Description
Backup-20140820-200630	Wed Aug 20 20:06:55 2014 UTC	Automatic backup: Writing backup file to CAStor failed.
Backup-20140820-201157	Wed Aug 20 20:12:21 2014 UTC	Automatic backup: Writing backup file to CAStor failed.

Copyright Caringo, Inc. All rights reserved.

Backup Domain

The Swarm storage domain to which the CSN writes configuration manifests and backups is configurable to allow administrators in tenanted clusters to determine where the backups are stored. The CSN will not create the domain if it does not exist so administrators must ensure the configured domain exists in the Swarm cluster or the domain configuration will fail in the console.

> Important
>
> If you are using Content Gateway (CloudScaler) and CSN together in the same environment, you must first create a domain in the Content UI Overview for your CSN backups and then specify this domain in the CSN Console.

Backup Manifest

A manifest of all saved backups is created in the storage cluster with the first backup set and updated with every subsequent backup. This manifest allows an administrator to retrieve the list of available backups from the storage cluster in the event of a CSN hardware failure. The UUID for the manifest is displayed at the top of the backup interface. It will not display until after the 2nd backup has been created. Note: Administrators should copy this UUID into a safe location so that in the event of a complete system failure the list of backup sets can be retrieved from the storage cluster. The backup manifest UUID is written with metadata that allows retrieval from an Elasticsearch-enabled cluster if the UUID is ever lost. Specifically the two following headers will be present on the manifest anchor stream:

```
CAStor-Application: backup_csn
```

```
CAStor-Application-Component: manifest
```

Automatic Backups

The backup utility watches a pre-determined list of CSN configuration files for file metadata changes. When a change is detected, the utility waits until the changes stabilize to prevent multiple incremental backups in a short time period and then creates a gzipped tar file with a complete set of all designated configuration files. This tar file is staged locally and then written to the storage cluster for protection. If for any reason, a backup fails to write to Swarm, an error will be logged and the backup will be retried periodically. Errors if the Swarm cluster is not fully online are expected; only persistent errors with a cluster present are cause for concern. The UUID of the backup set is also written into the Backup Manifest but is not displayed in the user interface. If the Backup Manifest is not retrievable for any reason, the backup set is written with metadata that allows retrieval from the Swarm cluster using Elasticsearch. Specifically, the following two headers will be present:

```
CAStor-Application-Automatic-Backup: yes
```

```
CAStor-Application-Component: saveset
```

The backup utility will periodically purge backups based on both age and backup count. The utility will keep at least 20 backups and, if there are more than 20 backups, it will purge any that are more than 30 days old.

Note: All backup times are displayed in Universal Time (UTC) and may not, therefore, correspond with the local system clock.

Manual Backups

Before you perform normal maintenance on the CSN, you can manually back up the configuration. You can manually back up the CSN configuration as often as every 30 seconds. Except for the fact that they are manually initiated, manual backups are identical to automatic backups with the exception that they do not contain the CAStor-Application-Automatic-Backup: yes header. To create a manual backup set at any time, click Create Backup on the Cluster Services > Backup and Restore page. In the Description field, enter a name to describe the backup and click Create Backup. The backup runs in the background until

Restoring a Backup

To restore the service configuration files and enabled/disabled status as they existed as part of a specific backup set, select the radio button next to the desired backup set and click the 'Restore Backup' button. This will restore all configuration files and each service's status (enabled or disabled) to their saved state at the time of the selected backup and reboot the server to the previous configuration. Following a successful reboot, administrators should immediately restart their Swarm cluster to ensure the networks are aligned and the node IP addresses are maintained.

After upgrade, previous backup sets may be marked as being incompatible with a previous software version if the backup format or system layout has been changed. These backups cannot be restored with the current version but will remain available if the software is reverted to a previous version using the csn-reset functionality.

> Note
>
> Due to breaking changes in the cluster.cfg EC settings for Swarm 8.1, CSN 8.1 and higher prevent you from restoring CSN backups from versions prior to 8.1.

Restoring Swarm UI: The backup/restore mechanism does not include the changes created by the `/usr/bin/configure_storage_webui_with_serviceproxy.py` script. To restore a CSN 8.3 that has run this script previously, perform these steps after the normal restore procedure.

1. Unset the target. Unsetting the target after a restore prompts Swarm to initialize Elasticsearch with the correct mappings. Use SNMP to set the configuration parameter `metrics.target` (MIB: `metricsTargetHost`) to a null value. Without this step, the Swarm cluster will continue to try to send metrics data to the configured Elasticsearch target, which is not yet available on the newly restored system. When Metrics are initiated for the first time using the metrics.target command, the proper mappings are made.
2. Run the setup script. Run /usr/bin/configure_storage_webui_with_serviceproxy.py on the newly restored CSN. This will setup the Swarm UI, service proxy, and metrics from scratch.

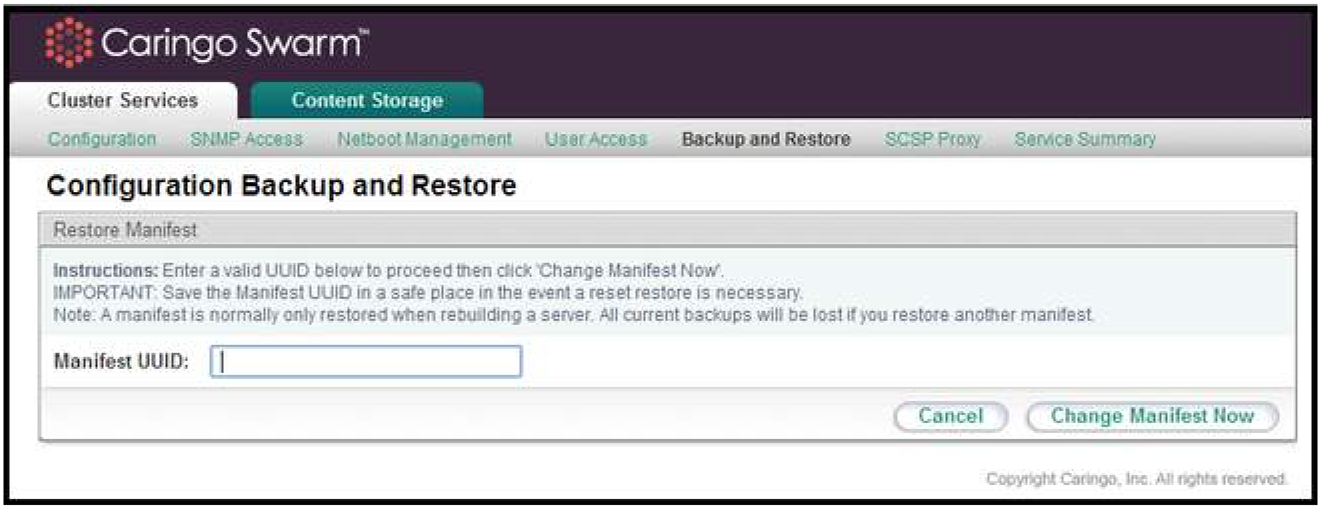
> Note
>
> When you restore to a fresh CSN, all previous metrics information stored in Elasticsearch before the restore process will be lost.

Failover

If a primary CSN fails, an administrator can choose to promote a secondary CSN to the primary CSN role by restoring the primary's backup manifest UUID onto the secondary and then restoring a backup set from the restored manifest list. The primary's backup manifest must have at least 2 backups in it prior to being used for failover. To assist with this transition, the secondary CSN periodically pulls the primary's Backup Manifest UUID via the privileged SSH channel and stores it in the following location on the secondary: / etc/caringo/csn/primary-manifest.txt. A timestamp on this file will notate the last time it was updated.

Note: If you are failing over a single-network primary to a single-network secondary, you must first configure the secondary's network to match the primary's previous configuration, including its IP address, since the CSN does not backup or restore network state for a single-network configuration.

To restore a manifest, click the 'Change Manifest' button at the top of the backup interface. This will bring up a entry box where the UUID you would like to restore can be entered. The entered UUID must be for a valid backup manifest created by the backup utility. If restoring a manifest on a machine that has an existing manifest and associated backups, you must be aware that the backup list will be completely overwritten when the entered manifest is restored.



Configuration Backup and Restore

Restore Manifest

Instructions: Enter a valid UUID below to proceed then click 'Change Manifest Now'.
IMPORTANT: Save the Manifest UUID in a safe place in the event a reset restore is necessary.
Note: A manifest is normally only restored when rebuilding a server. All current backups will be lost if you restore another manifest.

Manifest UUID:

Administrators should be aware that the secondary will effectively take on the network identify of the primary (both the cluster IP Address and the Primary's own IP Address) when the manifest and a selected backup within it are restored.

Note: Demotion of a Secondary CSN's backup set onto a Primary CSN is not supported. Failover should only be done when the Primary is not expected to return to service

Failover Without Swarm

If the failure of the Primary CSN coincides with an outage of the Swarm cluster, you will be unable to pull the Primary's Backup Manifest UUID from the cluster to restore it onto the Secondary. In this scenario, an administrator can manually restore the Primary CSN's last recorded backup set, which is updated hourly on the Secondary CSN if it has changed. The following command will restore the Primary's backup set onto the Secondary, effectively making the Secondary assume the role of the Primary CSN. The script should only be performed from the Secondary CSN with both the Primary CSN and the Swarm cluster offline:

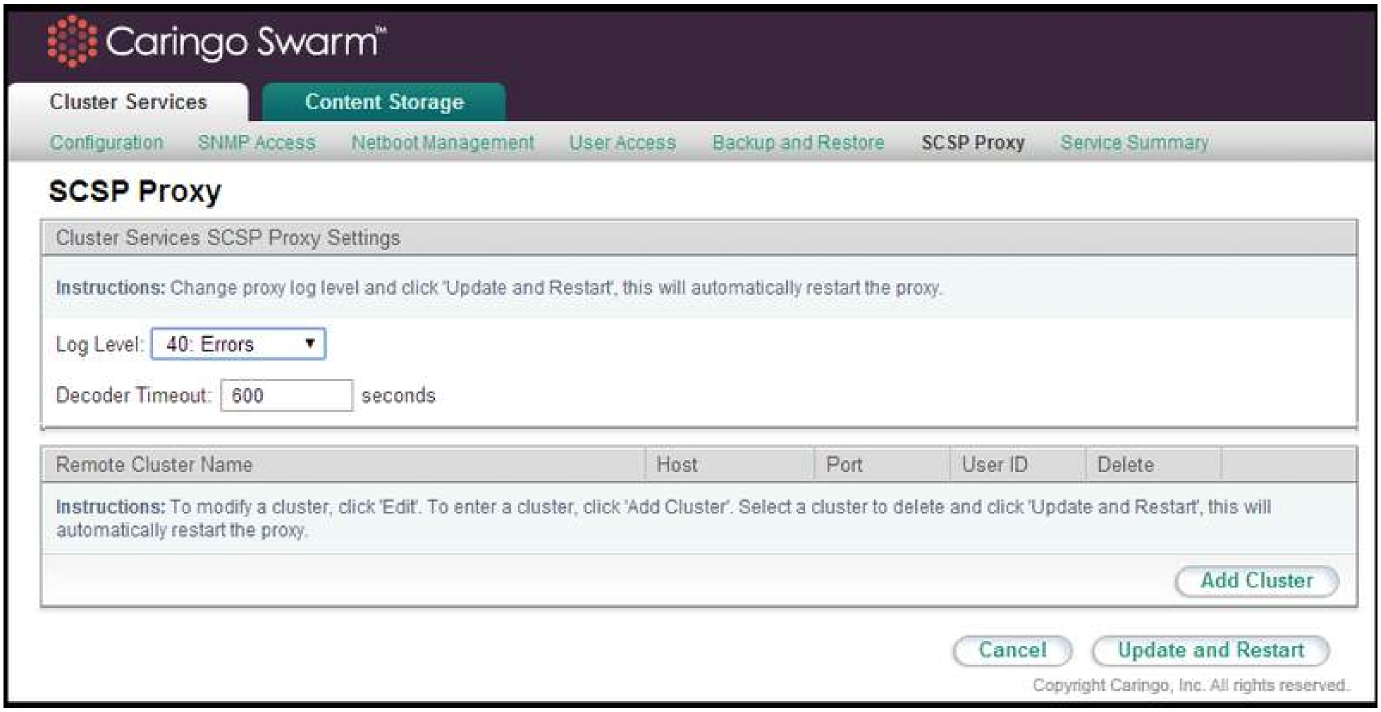
```
/opt/caringo/csn/bin/failover_without_castor
```

The script will restart the Secondary CSN after the Primary's configuration has been restored.

SCSP Proxy (dual-network configuration)

The SCSP Proxy installation on the CSN allows for clients who cannot directly address the Swarm cluster due to dual-network configuration to send and receive SCSP requests to and from Swarm nodes. The SCSP Proxy is configured during the network bootstrap process to listen on port 80 of both the external and internal interfaces in a dual-network configuration.

The log level for the SCSP Proxy can be changed from the user interface by selecting an alternate value from the list and clicking Update. After you change the log level, services restart after a few seconds.



SCSP Proxy

Cluster Services SCSP Proxy Settings

Instructions: Change proxy log level and click 'Update and Restart', this will automatically restart the proxy.

Log Level: 40: Errors ▾

Decoder Timeout: 600 seconds

Remote Cluster Name	Host	Port	User ID	Delete

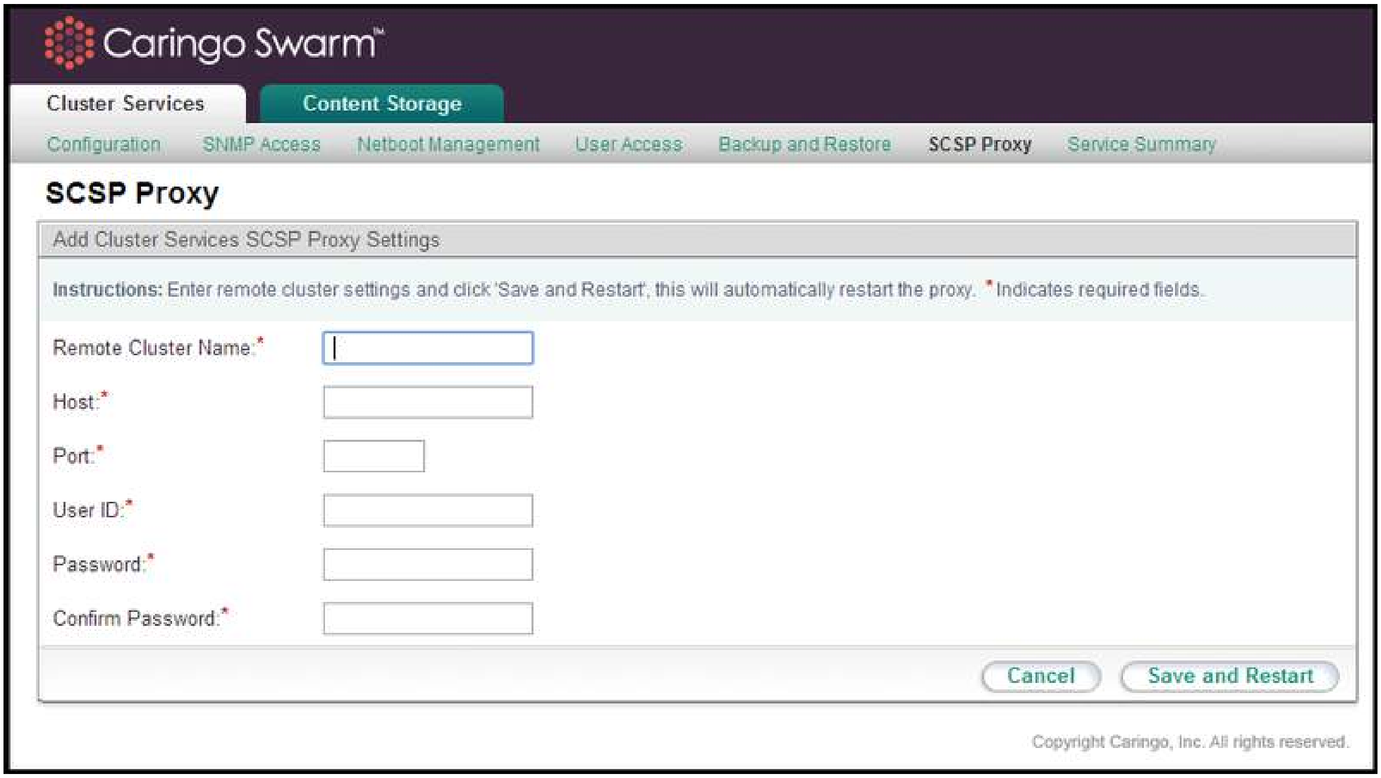
Instructions: To modify a cluster, click 'Edit'. To enter a cluster, click 'Add Cluster'. Select a cluster to delete and click 'Update and Restart', this will automatically restart the proxy.

Copyright Caringo, Inc. All rights reserved.

The SCSP Proxy is also capable of communicating with remote clusters to perform operations like global infos for one or more objects. See the SCSP Proxy Overview for a full overview of the supported remote methods.

To add a remote cluster to the configured list, click Add Cluster. To edit an existing remote cluster, click Edit. To delete a cluster from the list, select the check box in the Delete column next to the cluster to delete and click Update and

Restart.



SCSP Proxy

Add Cluster Services SCSP Proxy Settings

Instructions: Enter remote cluster settings and click 'Save and Restart', this will automatically restart the proxy. * Indicates required fields.

Remote Cluster Name:*

Host:*

Port:*

User ID:*

Password:*

Confirm Password:*

Copyright Caringo, Inc. All rights reserved.

The following information is required for all configured remote clusters:

Parameter	Description
Remote Cluster Name	The alpha-numeric name for a remote cluster that will be used in requests sent to the SCSP Proxy. Cannot contain whitespace. Can be the DNS name for the cluster but that is not required.
Host	The IP address or hostname of the remote SCSP Proxy the local SCSP Proxy will communicate with. Cannot contain whitespace.
Port	The port on which the remote SCSP Proxy will listen to incoming requests. This is usually port 80. May not contain preceding or trailing whitespace.
User ID	The name of an administrator that belongs to the Swarm Administrators group for the remote cluster. May not contain whitespace.
Password/Confirm Password	The password for the administrator specified in the User ID field. This value may contain spaces as long as they are not leading or trailing.

The CSN Console should be used for all configuration updates, but, for reference purposes, all SCSP Proxy configuration parameters are stored in the configuration file:

- `/etc/caringo/scspproxy/scspproxy.cfg`

- `/etc/caringo/scspproxy/hosts.cfg`

These files are automatically generated and should never be manually edited unless under direction of your support personnel. The SCSP Proxy service will be started automatically after the initial configuration of the CSN. If needed, it can be started, stopped or restarted as follows:

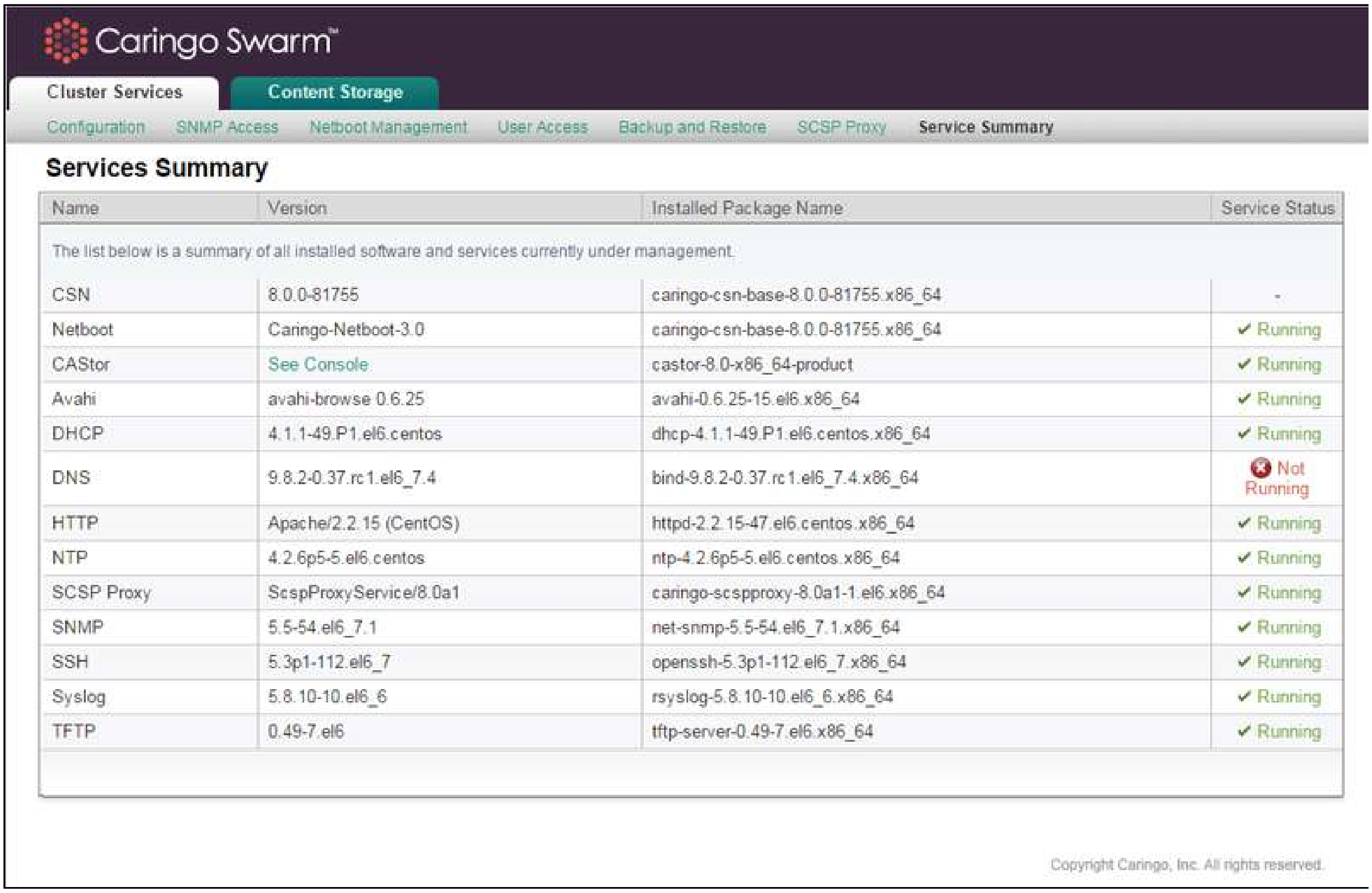
```
service scspproxy {start|stop|status|restart}
```

Services Summary (versions and status)

A list of all services that are important to the overall health and operation of the CSN is available on Cluster Services > Service Summary. When available, each service's running version, installed package name, and status are included for quick visibility into system state. Because Swarm nodes can run different versions, the version field for the Swarm software is linked to the storage console where the versions can be viewed on the nodes themselves.

If the installed package name is different than the listed version, the service in question might need to be restarted.

If a service is shown as not running, this does not necessarily indicate an error because some services are optional.



Services Summary

Name	Version	Installed Package Name	Service Status
The list below is a summary of all installed software and services currently under management.			
CSN	8.0.0-81755	caringo-csn-base-8.0.0-81755.x86_64	-
Netboot	Caringo-Netboot-3.0	caringo-csn-base-8.0.0-81755.x86_64	✓ Running
CAStor	See Console	castor-8.0-x86_64-product	✓ Running
Avahi	avahi-browse 0.6.25	avahi-0.6.25-15.el6.x86_64	✓ Running
DHCP	4.1.1-49.P1.el6.centos	dhcp-4.1.1-49.P1.el6.centos.x86_64	✓ Running
DNS	9.8.2-0.37.rc1.el6_7.4	bind-9.8.2-0.37.rc1.el6_7.4.x86_64	✖ Not Running
HTTP	Apache/2.2.15 (CentOS)	httpd-2.2.15-47.el6.centos.x86_64	✓ Running
NTP	4.2.6p5-5.el6.centos	ntp-4.2.6p5-5.el6.centos.x86_64	✓ Running
SCSP Proxy	ScspProxyService/8.0a1	caringo-scspproxy-8.0a1-1.el6.x86_64	✓ Running
SNMP	5.5-54.el6_7.1	net-snmp-5.5-54.el6_7.1.x86_64	✓ Running
SSH	5.3p1-112.el6_7	openssh-5.3p1-112.el6_7.x86_64	✓ Running
Syslog	5.8.10-10.el6_6	rsyslog-5.8.10-10.el6_6.x86_64	✓ Running
TFTP	0.49-7.el6	tftp-server-0.49-7.el6.x86_64	✓ Running

Copyright Caringo, Inc. All rights reserved.

Syslog Configuration

The initial configuration process, the CSN, SCSP Proxy and Swarm are all configured at install to log to the syslog server on the local CSN. Each application logs to a different log facility so the syslog server can separate the incoming log messages by product/function. The resulting log files are all located in /var/log/caringo:

- csnconfig.log

- castor.log
- csn.log
- scspproxy.log

All log files are rotated regularly and compressed. The logrotate process checks hourly to see if the log files need to be rotated based on either a size > 512mb or a created date > 1 week.

CSN Support Data Collection

To assist in troubleshooting field issues, the CSN supports the bulk collection of system state, configuration and logs via integration with a standard RHEL/CentOS tool, sosreport. To initiate data collection, simply type sosreport at the command line. You will be prompted to enter your name and a support case number. Your name will be defaulted to the name used to register your RHEL version. If you do not have a case number, enter any value. The tool will then collect relevant data about the system and combine it into a tar file that is saved to the /tmp directory with a generated MD5 for later validation. When necessary, this tar file should be sent to your support representative as one of the first steps in troubleshooting any system issues. The size of the tar file may be as large as 25MB depending on the amount of logging present on the system.

Console - Content Storage

This section describes how to run and manage the network and configuration services in a single Swarm cluster using the Content Storage tab options in the CSN console. The CSN currently provides network and configuration services for a single Swarm cluster.

> Note
>
> CSN 8.3 includes both the Swarm Storage UI and Swarm Metrics, which populates the UI with usage data. With these new cluster reporting mechanisms, the legacy Reporter tab has been removed.

Configuration (storage cluster)

The CSN will automatically configure Swarm in multi-server mode to take advantage of all available CPU cores. The number of Swarm processes assigned to each node is dynamically determined based on system information provided by the node when it requests its network boot configuration. IP addresses are statically assigned to each node and then maintained by a CSN process that pings each node every 5 minutes. If a node has not responded to a ping in 72 hours, its IP address will be released and a new one assigned if the node should come back online at a later time. The maximum number of allowed processes per node, the interval at which nodes are pinged (default 5 minutes) and the length of time the CSN waits before releasing the IP address for a node that has not responded (default 72 hours) are all configurable. Please contact your support representative for instructions on modifying these parameters.

Note: When replacing older hardware, particularly if running with a small network, you may need to either work with your support representative to decrease the IP release timeout or bring new nodes online one at a time to ensure you have enough IP addresses available until the older nodes have released their IPs after the timeout period.

A list of all IP address assignments and the associated node's MAC address can be obtained by running the following command line script:

```
/opt/caringo/csn/bin/ip-assignments
```

MAC addresses will appear multiple times in the output if the nodes have enough CPU cores to support more than one Swarm process. The script will not return any results if no storage nodes have been booted yet.

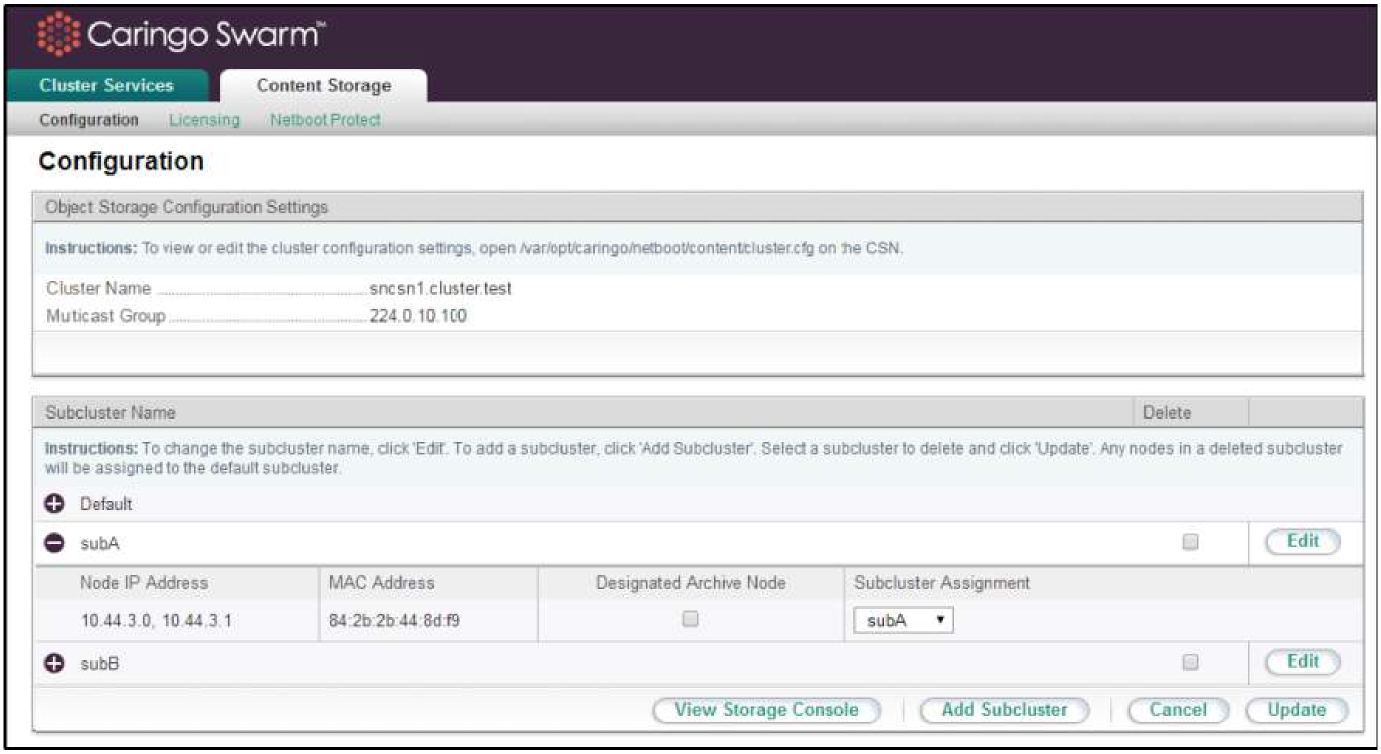
By default, the configuration passed to Swarm nodes during the netboot process will specify that all network interfaces be bonded using Balance-ALB bonding. Nodes must be configured to boot in BIOS mode; UEFI booting is not currently supported from the CSN.

Booting Nodes

Dual-network	Upon boot of a new node onto the internal network via network boot, a CSN configured in dual-network mode will automatically provide the needed configuration and software. With subsequent boots of a previously known node on the network, the CSN will recognize the Swarm node and use its previously assigned configuration to boot the node. For an additional level of control over what servers get booted as Swarm nodes in a dual-network configuration, administrators may choose to enable netboot protection as described below. To enable netboot protection, click the 'Enable Protection' button at the bottom of the Content Storage > Netboot Protect page of the CSN console.
Single-network	Once a server has its MAC addresses listed on the Content Storage > Netboot Protect page, you may boot the server and the CSN will provide the needed configuration and software. If your single-network environment is controlled to the point where booting unknown nodes is not a concern, you may alternately disable netboot protection by clicking the Disable Protection button. This will allow any server that requests a netboot image to be formatted as a Swarm node without having its MAC addresses listed. Important Unlike a dual-network configuration, where the CSN is the network gateway for the Swarm nodes, a single-network configuration has the Swarm nodes using the same gateway as the CSN.

Updating Configuration Settings

In addition to the default network configuration information gathered from the initial configuration of the CSN, the CSN Console displays a few configuration parameters from the Content Storage > Configuration page.



Configuration

Object Storage Configuration Settings

Instructions: To view or edit the cluster configuration settings, open /var/opt/caringo/netboot/content/cluster.cfg on the CSN.

Cluster Name sncsn1.cluster.test
Muticast Group 224.0.10.100

Subcluster Name | Delete

Instructions: To change the subcluster name, click 'Edit'. To add a subcluster, click 'Add Subcluster'. Select a subcluster to delete and click 'Update'. Any nodes in a deleted subcluster will be assigned to the default subcluster.

Node IP Address	MAC Address	Designated Archive Node	Subcluster Assignment
10.44.3.0, 10.44.3.1	84:2b:2b:44:8d:f9	□	subA ▾

Field	Description
Cluster Name	The read-only name of the cluster that was specified during the initial network configuration process.
Multicast Group	The multicast address for the cluster. It must be a Class D IP address in the range 224.0.0.0 - 239.255.255.255.

See Persisted Settings, which describes runtime settings that must be changed via SNMP.

To make changes to boot-time settings on the CSN, edit the configuration file located at `/var/opt/caringo/netboot/content/cluster.cfg`. Any boot- time configuration changes, both cluster and node level, will not take effect until the cluster is rebooted to pick up the new configuration. Note that manually configuring node IPs is not supported as the CSN expects ownership of the IP address space for Swarm nodes both when assigning IPs for new nodes and when gathering data for reporting.

The Swarm configuration page allows editing of a few node-level parameters. All nodes that have been network booted by the CSN will be included in the listed nodes with their MAC address and IP address. Any nodes that previously booted from the CSN but have since expired will not be included. For multi-process servers, each row in the interface will contain multiple IP addresses that identify the different virtual nodes in the server. These virtual nodes cannot have their subcluster or archive node status set independently. The nodes will be listed underneath their assigned subcluster. If no subcluster has been assigned, nodes will be listed in the 'Default' subcluster.

From the node list, it is possible to set the subcluster for an individual node, as well as whether or not it should be set as an archive node. See the Node Configuration for usage information of these configuration parameters. Prior to assigning subclusters to a node, the list of allowed subclusters must be created using the 'Add Subcluster' button to ensure consistency. Subcluster names may only contain letters, numbers, spaces, and underscores. If a subcluster is edited or deleted, all corresponding entries for each related node will also be updated.

Using the Swarm Admin Console

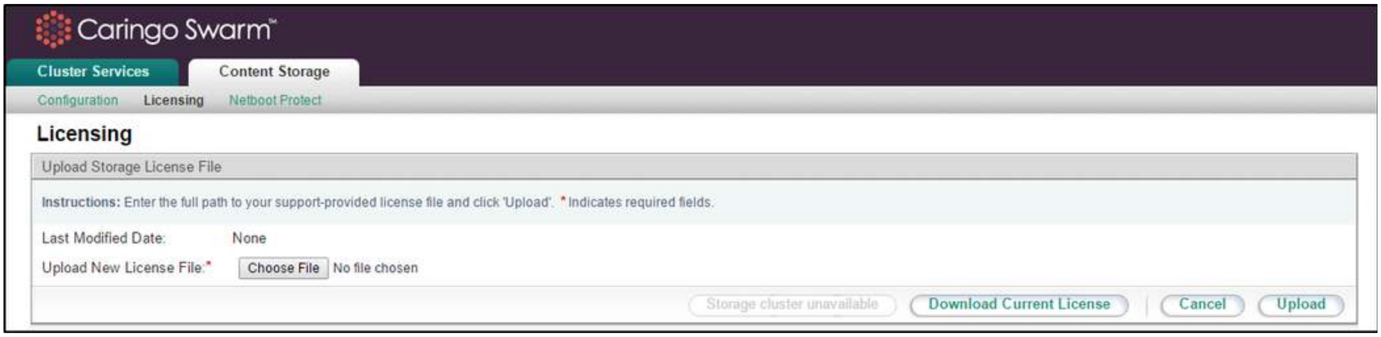
Swarm includes its own administrative console, which can be reached directly from the external network by entering the following address:

```
http://{CSN·external·IP}:8090/services/storage/
```

The Swarm admin console may also be accessed by clicking the View Storage Console button on any page of the Conte nt Storage tab page of the CSN Console.

Licensing (upload license file)

Use the Licensing tab to update your Swarm license file whenever there are changes:



Licensing

Upload Storage License File

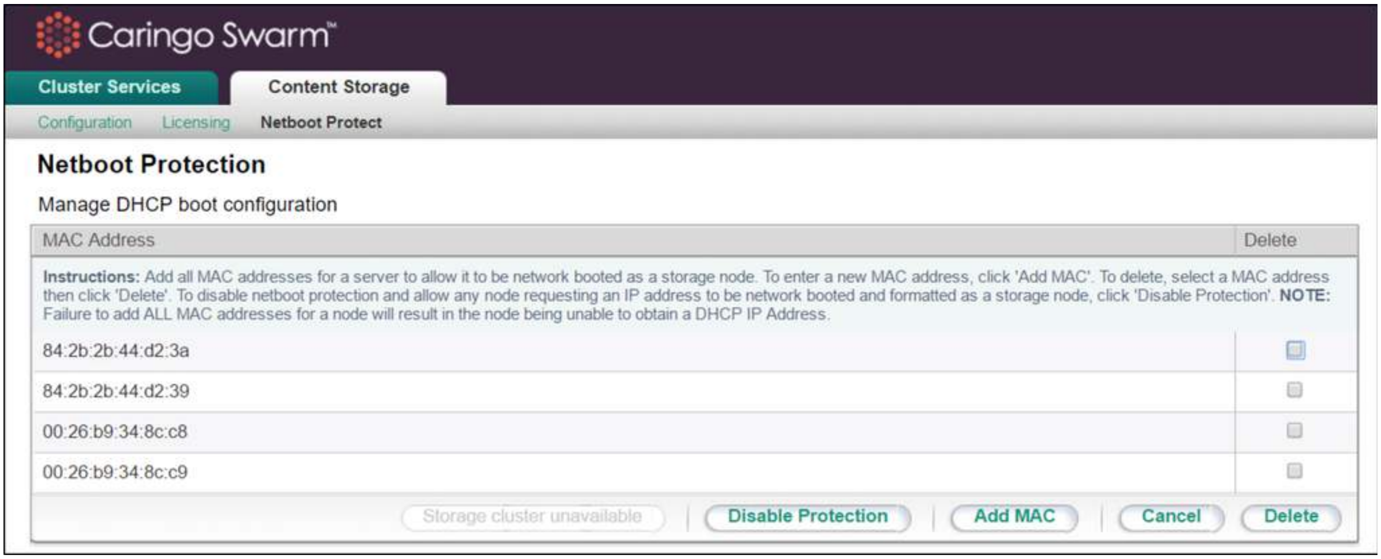
Instructions: Enter the full path to your support-provided license file and click 'Upload'. * Indicates required fields.

Last Modified Date: None

Upload New License File:* Choose File No file chosen

Netboot Protect (from unintended formatting)

To prevent the unintended formatting of servers not intended to be Swarm nodes when netboot protection is enabled (default for single network installations), you must first add all MAC addresses for any server you do intend to use as a Swarm node on the Content Storage > Netboot Protect page of the CSN console.



Netboot Protection

Manage DHCP boot configuration

Instructions: Add all MAC addresses for a server to allow it to be network booted as a storage node. To enter a new MAC address, click 'Add MAC'. To delete, select a MAC address then click 'Delete'. To disable netboot protection and allow any node requesting an IP address to be network booted and formatted as a storage node, click 'Disable Protection'. **NOTE:** Failure to add ALL MAC addresses for a node will result in the node being unable to obtain a DHCP IP Address.

MAC Address	Delete
84:2b:2b:44:d2:3a	□
84:2b:2b:44:d2:39	□
00:26:b9:34:8c:c8	□
00:26:b9:34:8c:c9	□

Due to NIC bonding, entering a single MAC address for a server is not sufficient; all MAC addresses must be included. If you are unsure how to obtain the MAC addresses for the server, try booting the storage nodes and then check the /var/log/messages log to see what MAC addresses requested an IP address. You can search for related DHCP messages in that log with a command similar to the following:

```
[root@csninternalnic ~]# grep 'DHCPDISCOVER' /var/log/messages
```

A matching log message might look like the following, where e0:91:f5:08:6b:17 is the MAC address of a node requesting an IP address:

```
127.0.0.1 <27>Oct 9 16:21:05 dhcpd: DHCPDISCOVER from e0:91:f5:08:6b:17 via bond0: network
10.44.0.0/16: no free leases
```

The 'no free leases' error message is an indication that a node has asked for a DHCP address but it is not included in the netboot protection page so an IP Address is not granted. This method of finding MAC addresses may result in extra results if there are servers that are not Swarm nodes mistakenly asking the CSN's DHCP server for an IP Address.

> Warning
>
> If you enable netboot protection but do not add any MAC addresses to the list, you will not be able to boot any Swarm nodes.

Swarm Storage Cluster

This section describes how to configure and manage a Caringo® Swarm™ storage cluster using the Swarm Management UI. It covers how a Swarm administrator can:

- Configure the cluster nodes
- Manage the cluster and nodes
- Manage volumes
- Implement multiple nodes in a single server
- Integrate a storage cluster into a corporate enterprise
- Upgrade a license or cluster
- Troubleshoot a storage cluster

> Proactive support
>
> Swarm clusters send basic configuration, usage, and health information reporting to Caringo once a day from every node in the cluster. No user-stored data is included, and this information is encrypted prior to transmission. Contact your support representative with any questions related to this support functionality.

Swarm Concepts

Caringo Swarm is unified storage software that leverages simple and emergent behavior with decentralized coordination to handle any rate, flow, or size of data. Swarm turns standard hardware into a scalable, highly available pool of storage resources that adapts to any workload or use case while offering a foundation for new data services.

This section introduces fundamental concepts that you will need to make full use of Swarm:

- How Swarm interacts with its content objects
- How you access objects and leverage feeds
- How you protect and secure your content
- How you store large objects efficiently
- How you use versioning in Swarm
- Understanding Swarm Objects
- Working with Swarm Objects
- Understanding Feeds
- Elastic Content Protection
- Working with Large Objects
- Version Control for Objects
- Swarm Metrics

Understanding Swarm Objects

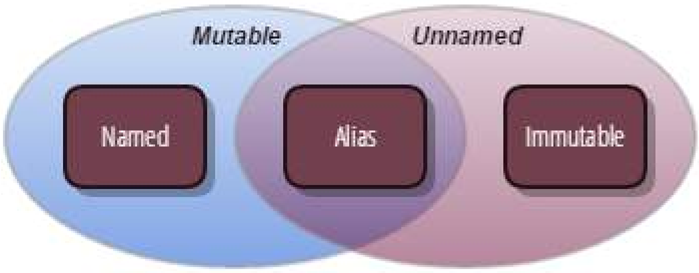
- Types of Data Objects
- Types of Container Objects
- Understanding Unnamed Objects
- Understanding Named Objects

Types of Data Objects

All data objects used by Swarm are either unnamed or named.

- Unnamed objects are assigned identifiers by Swarm. They are created, updated, accessed, and deleted using an unambiguous UUID. Swarm is optimized to handle unnamed objects most efficiently, and the random assignment of UUIDs to content offers a higher level of programming security.
- Named objects are assigned identifiers by you. They are created, updated, accessed, and deleted using the name that you chose. You can create multiple objects with the same name as long as they are in different buckets. Because these objects require a name lookup, they cannot perform as well as unnamed objects.

Swarm supports three types of data objects: Named, Alias, and Immutable. Two of the three you can update (are mutable) and two of the three you identify by UUID (are unnamed). All Swarm objects are one of these three types, shown as the dark boxes:



- Immutable (unnamed) objects. Immutable objects have UUIDs and content that will never change. If you delete an immutable object, Swarm retires its UUID as well so that it will not be reused for another object. Using a special header, you can lock an immutable object to prevent its deletion, which lets you implement Write Once Read Many (WORM) (see Lifepoint Metadata Headers).
- Alias (mutable) objects. Alias objects have permanent UUIDs but replaceable content. Like immutable unnamed objects, an Alias object's UUID is retired after you delete the object.
- Named objects. Named objects have replaceable content and are accessed by name only. If you delete a named object, you are free to create another object with the same name later.

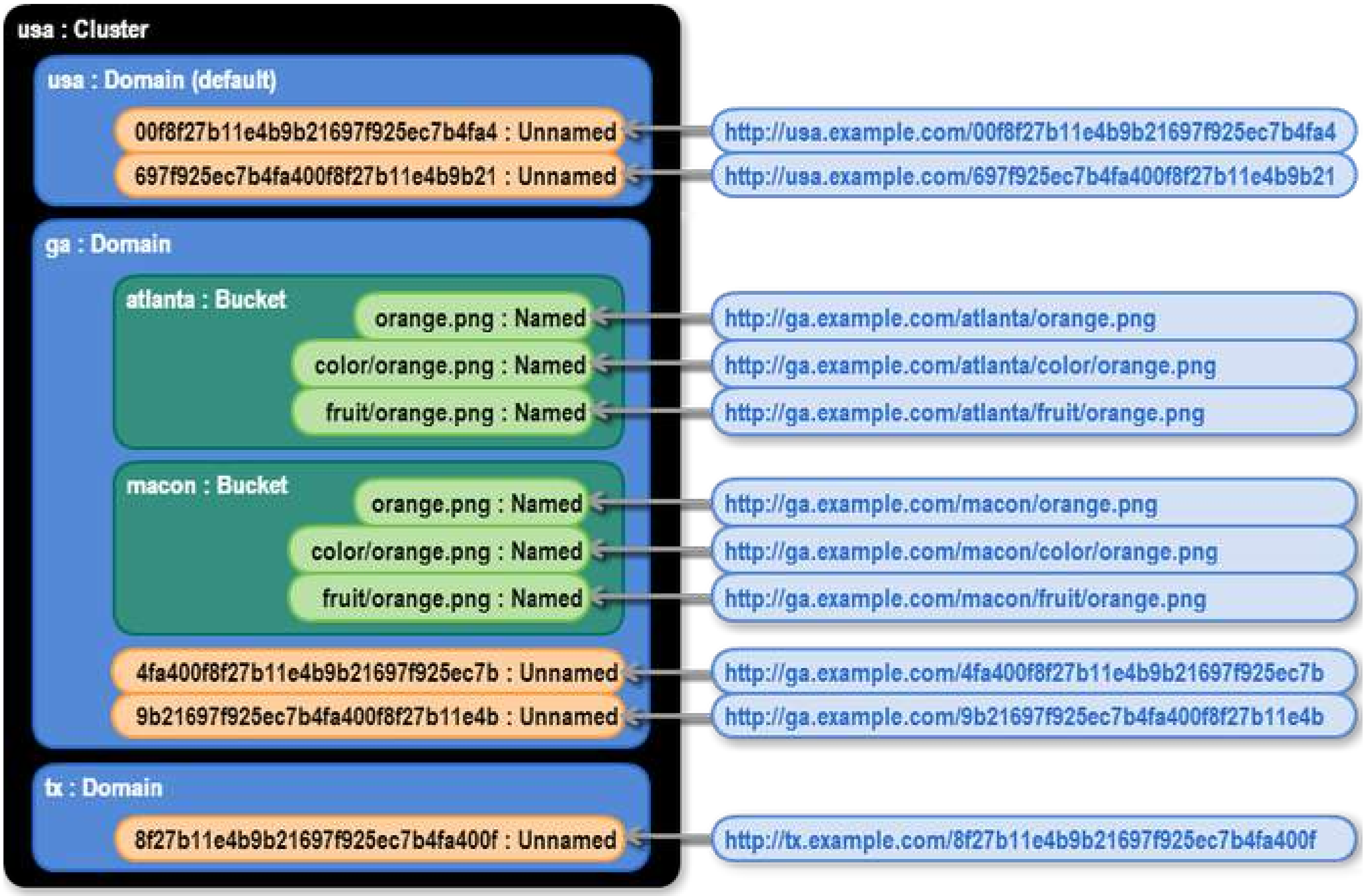
To manage objects with Swarm, applications communicate using the Simple Content Storage Protocol (SCSP). SCSP methods and syntax are a proper subset of the HTTP/1.1 standard. The same SCSP methods apply to all of these types of objects.

> See Using SCSP.

Types of Container Objects

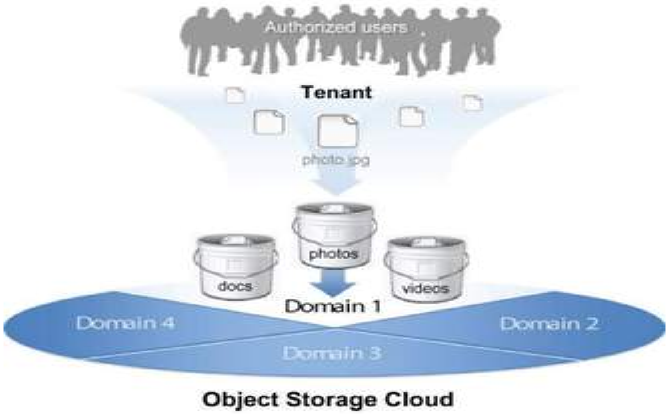
- Clusters
- Domains
  - Best practices
- Buckets

To let you organize storage of your data objects, Swarm provides a hierarchy of container objects: cluster, domain, and bucket (shown in white, below).


usa : Cluster
usa : Domain (default)
00f8f27b11e4b9b21697f925ec7b4fa4 : Unnamed
697f925ec7b4fa400f8f27b11e4b9b21 : Unnamed
http://usa.example.com/00f8f27b11e4b9b21697f925ec7b4fa4
http://usa.example.com/697f925ec7b4fa400f8f27b11e4b9b21
ga : Domain
atlanta : Bucket
orange.png : Named
color/orange.png : Named
fruit/orange.png : Named
http://ga.example.com/atlanta/orange.png
http://ga.example.com/atlanta/color/orange.png
http://ga.example.com/atlanta/fruit/orange.png
macon : Bucket
orange.png : Named
color/orange.png : Named
fruit/orange.png : Named
http://ga.example.com/macon/orange.png
http://ga.example.com/macon/color/orange.png
http://ga.example.com/macon/fruit/orange.png
4fa400f8f27b11e4b9b21697f925ec7b : Unnamed
9b21697f925ec7b4fa400f8f27b11e4b : Unnamed
http://ga.example.com/4fa400f8f27b11e4b9b21697f925ec7b
http://ga.example.com/9b21697f925ec7b4fa400f8f27b11e4b
tx : Domain
8f27b11e4b9b21697f925ec7b4fa400f : Unnamed
http://tx.example.com/8f27b11e4b9b21697f925ec7b4fa400f

Clusters

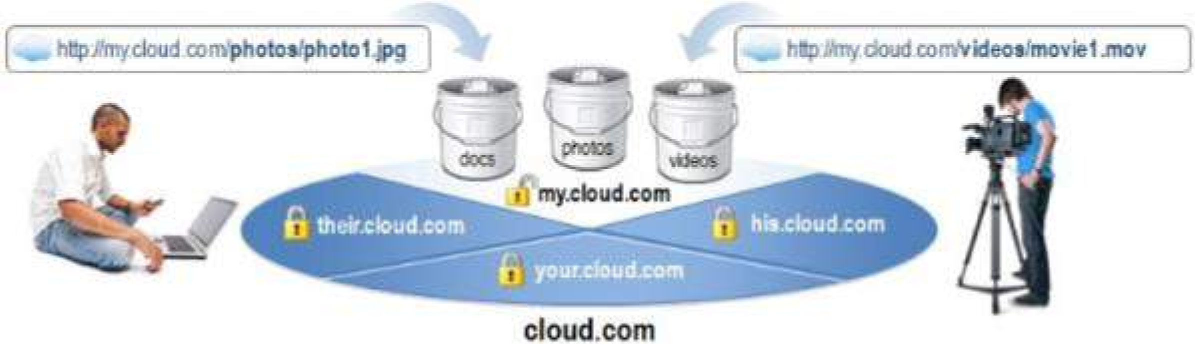
The cluster is the top-level container in your object storage cloud. A cluster can have several storage domains, each organized with its own buckets:



These storage domains and buckets can enforce many business requirements:

- Assign permissions
- Issue metadata search or listing requests
- Account for usage and billing

You can configure your cluster into a storage cloud (such as cloud.com, below) that lets users in one storage domain (my.cloud.com) securely access their content separately from other users:



Domains

A domain is the highest-level container in which you place data objects, and it is the primary way to control access to and grouping of the data.

A cluster administrator creates the domain either using the Swarm Admin Console or programmatically, with an SCSP command. Buckets and data objects must be created in a domain (tenanted), so plan to create your domains first.

Best practices

These actions make your URLs and cURL commands as simple as possible:

- Have your DNS map the cluster name to one or more of your Swarm nodes.
- Create a storage domain that matches the name of the cluster. This is the default domain if no domain is specified in the SCSP request or if the specified domain does not exist.

> See the Naming Rules.

Buckets

Buckets are the required organizing containers for any named objects in a domain. Buckets let you create logical paths to named objects, and they provide a layer of access control.

Each bucket operates as a subdirectory in the domain:

```
MYDOMAIN/my_bucket/my_document.doc
```

Note

Buckets are not folders: they cannot be nested in other buckets.

Since named objects only exist within buckets, the bucket must be created prior to storing a named object.

Important

Never write named objects without specifying a bucket (with a path like `http://cluster.example.com/testfile.txt`): doing so creates the object as a bucket. Buckets are treated differently, and they are not meant to contain a content body.

When you create a new bucket, the cluster synchronously creates two replicas of the bucket object and then asynchronously creates any additional replicas as determined by a lifepoint header or the scsp.defaultContextReplicas and scsp.maxContextReplicas configuration parameters.

> See the Naming Rules.

Understanding Unnamed Objects

Universally Unique Identifiers

When you create an unnamed object, Swarm assigns it a unique identifier that is different from every other identifier assigned to every other unnamed object ever stored, past and future. This identifier is known as the object's Universally Unique Identifier (UUID).

> Tip
>
> Think of a UUID like a coat check ticket. When you check a coat at a restaurant, you receive a coat check ticket that identifies you as the coat owner. To retrieve your coat when you are ready to leave, you present the ticket, not a description of the coat. No ticket, no coat.

A Swarm UUID is a sequence of 128 bits. In text-based languages and protocols such as Swarm's Simple Content Storage Protocol, a UUID is represented as a sequence of 32 hexadecimal digits:

```
http://companyname.example.com/12BFEA648C2697A56FD5618CAE15D5CA
```

A UUID has no internal structure and cannot be parsed in any way to yield information about where or when the associated object was stored. An object's UUID is not derived from its content.

UUIDs and Replicas

Swarm can produce copies (or replicas) of an object to facilitate fault-tolerance, integrity, or speed of access. Even so, every replica of a given object has exactly the same UUID as the original object. There is no external way to distinguish an original from a replica. For all practical purposes, consider each replica of an object completely identical to every other replica of that object in the cluster.

Immutable Objects

By default, once you store an unnamed object in Swarm, you can change it only by deleting it (if permitted by the object's lifepoint policy). There is no way to open an immutable object for updates or to extend an object by adding more data to the end of it.

To store an object in the cluster, your application must provide a size (in bytes) for the object. Swarm then allocates exactly enough space to store the given number of bytes. After that, every object replica has precisely the same byte size. There is no chance that a replica will be updated or changed in some way while others are not changed.

If your application needs to update an immutable object with new content, the application is responsible for:

- Storing a new Swarm object containing the updated data (or delete it if necessary)
- Modifying any references to the old UUID to point to the new UUID
- Maintaining association between the old object and the new revision

If those associations are important for your situation, consider using alias objects.

Unnamed Mutable (Alias) Objects

Alias objects are a special type of unnamed object in Swarm because you can replace the content and the UUID remains constant (or alias). An alias object is created in much the same way as a regular object but uses an ?alias quer y argument on the WRITE request. Unlike an immutable object, where the contents can never change, the contents of an alias object can be replaced using an SCSP COPY, APPEND, or UPDATE request. When reading an alias object, Swarm always returns the most recent data associated with the alias object object's UUID.

Alias objects serve a very specific purpose for applications that store fixed content data. Many such applications must associate a symbolic name of some kind (for example, a URI or a file path name) to an object UUID returned from Swarm.

> Note
>
> Named and alias objects can be updated at a maximum frequency of once per second. Updating more frequently can cause unpredictable results with the stored object version. If your application updates objects faster than once per second, include the ?replicate query argument to ensure that more than one node can return the latest version in a subsequent read.

Accessing Unnamed Objects

You access unnamed objects using their UUID, placing it at the end of the URI as a string of 32 case-insensitive hexadecimal digits:

```
http://companyname.example.com/12BFEA648C2697A56FD5618CAE15D5CA
```

This URI specifies three things:

- Protocol: `http`
- Cluster: `companyname.example.com`
- UUID: `12BFEA648C2697A56FD5618CAE15D5CA`

> Note
>
> The length of the UUID string must be exactly 32 characters, including any leading zeros.

Creating Unnamed Objects

When you store an object for the first time, the UUID is not yet assigned, so an SCSP WRITE request includes just the first two components of the URI. After the data is transferred and stored, Swarm generates and returns a new UUID to the storing application.

Be sure to use a HOST header equivalent to the cluster name, the host IP address, or a domain=clusterName query argument on all requests.

> Caution
>
> Ensure that your application is not passing a HOST header that is neither an IP address nor a domain that exists in the cluster (unless the host header matches the cluster name). Swarm will attempt to look up the non-existent domain on every request and will wait for multiple retries before the lookup times out, impacting performance.

Unnamed Object Changes in Version 6.0

Beginning with Swarm 6.0, you can create unnamed objects in a domain and apply the domain's protection settings to those objects. This allows unnamed objects to be accounted for in metered environments where storage may be allocated to a set of users based on the domain into which their content is stored. An object's association with a domain is called tenancy. However, the domain is not used to later locate the unnamed object in the cluster as is the case for named objects.

An SCSP WRITE of an unnamed object that should be associated with a domain can be formatted using the following syntax:

```
POST /?domain=domain-name[&alias]
POST /[?alias] Host: domain-name
```

> Important
>
> When writing unnamed objects, use a HOST header equivalent to the cluster name, the host IP address, or a domain=clusterName query arg on all requests even if you are not using domains for other purposes.

> Caution
>
> When writing unnamed objects, ensure that your application is not passing a HOST header that is neither an IP address nor a domain that exists in the cluster (unless the host header matches the cluster name). Swarm will attempt to look up the non-existent domain on every request and will wait for multiple retries before the lookup times out, impacting performance.

The alias query argument is required to create an alias object. To create an immutable object, simply omit the alias query argument.

The following table summarizes the differences with unnamed objects and domains in version 6.0 and later compared to earlier ones:

Unnamed Behavior in Version 6.0+	Behavior in Earlier Versions

An unnamed object can be created in a domain (but not in a bucket). An unnamed object created in version 6.0 and later is associated with a domain only if the object is explicitly associated with a domain using either the domain= query argument or a host header. If the configuration parameter cluster.enforceTenancy is set to false (default) and either a domain is not specified, the default domain is specified, or a domain that does not exist is specified on the POST using either the domain= query arguments or the host header, then the object is not associated with a domain. You cannot write unnamed streams into the default domain unless cluster.enforceTenancy=True If the configuration parameter cluster.enforceTenancy is set to true and an alternate domain is not specified on the POST request, then the object is associated with the default cluster-name domain. With cluster.enforceTenancy set to true, specifying a domain that does not exist on the POST request will result in a 412 response.	Unnamed objects were always untenanted (not associated with any domain).
Whether unnamed objects are permitted in a particular domain is determined by the protection settings created in the domain, including the default domain. If unnamed objects are not associated with a domain, there are no POST restrictions.	Whether unnamed objects POSTs were allowed was always determined by the protection settings of the default cluster domain.

> See WRITE for Unnamed Objects.

Understanding Named Objects

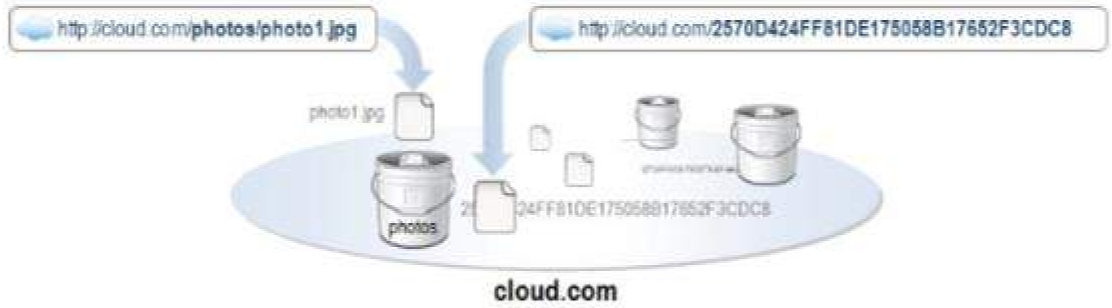
- Named versus Unnamed
- Accessing Named Objects
- Creating Named Objects
- Overwriting Named Objects
- Deleting Named Objects

Named objects give you a way to store and retrieve content by user-provided name, rather than by Swarm-assigned UUID. This structure makes it possible to use third-party protocols, such as S3.

A named object refers to a customer-named object (such as a video file) contained within a bucket. Using meaningful bucket and object names creates friendly, readable URLs to stored content.

Named versus Unnamed

The illustration below contrasts two data objects—one named and one unnamed—in a cluster domain named cloud.com. The named object is stored in a bucket named photos, and the unnamed object is identified by its UUID at the root of the domain:


http://cloud.com/photos/photo1.jpg
http://cloud.com/2570D424FF81DE175058B17652F3CDC8
photo1.jpg
photos
2570D424FF81DE175058B17652F3CDC8
cloud.com

Your Swarm private or public cloud can have whatever combination of object types you need:

- All unnamed objects
- All named objects (which must be assigned to buckets)
- Both named and unnamed objects in any proportion

Pathnames to named objects must be unique:

- Every object within a bucket must have a unique name.
- Every bucket within a domain must have a unique name.
- Two or more objects in the same cluster may have the same name, but only if they are in different buckets or in different domains.

Accessing Named Objects

You access a named object by combining two strings: a domain name and a pathname to the object, in this form:

```
{domain-name}/{bucket-name}/{object-name/which/can/have/slashes}
```

Important
Even though objects in buckets resembles a file system, these are not files in folders. There is no file system u nderlying the apparent path structure of named objects.

```
http://cluster.example.com/marketing/ads/about-object-naming.m4v
```

In this example,

- cluster.example.com is the domain name.
- marketing is the bucket name (buckets are required).
- ads/about-object-naming.m4v is the object name (ads/ is merely part of the name).

Note
An object name like /folder1/../folder2/object.txt is not the same as /folder2/object.txt . These legitimate names specify two different objects.

Creating Named Objects

Following are simple cURL examples for creating a bucket and creating named objects within that bucket.

To use these examples, you must have access to a domain that does not have its protection settings set to allow only a specific set of users POST privileges. If you have access to a development environment, create a domain with the following protection setting:

```
All Users. No authentication required.
```

> If you are not sure how to create a domain, contact your cluster administrator or see Manually Renaming a Domain or Bucket in a Mirrored or DR Cluster.

The following examples assume that the domain is named test.example.com and that you are sending commands to a node whose IP address is 172.16.0.35.

To create a bucket and objects, open a terminal window and execute the following:

1. Create a bucket. (replicate=immediate is the Replicate On Write option.)

```
curl -i --post301 --data-binary ''
  --location-trusted
'http://172.16.0.35/bucket?domain=test.example.com&replicate=immedi
ate'
  -D create-bucket.log
```

2. Create a named object in the bucket.

```
curl -i --post301 --data-binary '<html><h1>Hello world</h1></html>'

  -H 'Content-type: text/html'
  --location-trusted
'http://172.16.0.35/bucket/test.html?domain=test.example.com'
  -D create-object.log
```

3. Verify the object in a browser. Enter the following in the Address or Location field:

```
http://172.16.0.35/bucket/test.html?domain=test.example.com
```

4. Change the object by adding data to it.

```
curl -i --post301 -X APPEND --data-binary '...to be continued...'
  -H 'Content-type: text/html'
  --location-trusted
'http://172.16.0.35/bucket/test.html?domain=test.example.com'
  -D update-object.log
```

5. Refresh your browser to see the updated object.

Note

Named and aliased objects can be updated at a maximum frequency of once per second. Updating more frequently can cause unpredictable results with the stored object version. If your application updates objects faster than once per second, include the replicate query argument to ensure that more than one node can return the latest version in a subsequent read.

Overwriting Named Objects

Using the POST method and a simple HTTP request, you can overwrite a named object that currently exists in your storage cluster.

To prevent overwriting an existing object, include the If-None-Match: * request header.

- If the named object does not exist, Swarm WRITEs the named object.
- If the named object exists, Swarm responds with a 412 Precondition Fail error.

> See SCSP WRITE.

Deleting Named Objects

Swarm allows you to delete a bucket or domain that still contains content.

Important

Do not create orphaned content by deleting its container object.

Use one of these methods to avoid creating orphans:

- Delete the content first before you delete the bucket or domain.
- Add the recursive query argument when you delete the domain or bucket, which causes the health processor to remove asynchronously any content it finds in the deleted context.

If you do not, Swarm may generate replication and search indexer warnings in the syslog and Admin Console as it tries to access content that is missing its parent context. The logs will include error messages that allow you to see the alias UUIDs of the missing bucket or domain:

```
Domain 'example.com' (uuid=...) has been deleted with orphan content.
Consider recreating.
```

> See Restoring Domains and Buckets for how to delete orphaned content.

Working with Swarm Objects

- Naming Rules
- Universal Resource Identifiers
- Content Cache
- Data Protection

Naming Rules

Follow these rules for naming the objects that you create for storage in Swarm:

Type	Reference	Rules and Notes	Examples
Tenant	RFC 1034	Applies to Gateway only. A tenant must follow the naming rules of a domain.	

Domain	RFC 1034	For maximum compatibility, ensure that your domains are valid DNS names that resolve in your network.
		A domain name must
		• Be a 7-bit ASCII byte sequence.
		• Be case-insensitive.
		• Begin with an alphanumeric character.
		• Use alphanumeric characters, underscore (_), period (.), and hyphen (-).
		• Not have adjacent or final hyphens or periods (--, .., -., .-).
		• Not be an IPv4 or IPv6 IP address.
		• (S3 compatibility) Not be longer than 253 characters.
			Valid:
			`my-cluster.example.com`
			`my_cluster.example.com`
			Invalid:
			`domain`
			`cluster_example_com`

Bucket	RFC 1034	A bucket name (which is only used in the path) must • Be unique within the domain. • Be case-sensitive. • Be a valid URL-encoded, UTF-8 byte sequence. ◦ Content UI: URL encoding is taken care of by the user interface. • Not be a UUID (32 hexadecimal characters). • Not exceed 8000 characters (larger than that is not tested or supported). • (S3 compatibility) Use lowercase ASCII and DNS-compatible names not longer than 63 characters.	
Named object	RFC 3986	An object name must • Be unique within the bucket. • Be case-sensitive. • Be a valid URL-encoded, UTF-8 byte sequence. ◦ Content UI: URL encoding is taken care of by the user interface.	Valid: `Accounting/Customer23-03/15`

Note
While you may use non-ASCII characters (such as "`résumé.doc`") in bucket and object names, the URL must be properly escaped in the HTTP request ("r%C3%A9sum%C3%A9.doc").

Universal Resource Identifiers

- Supported Application Protocols
- Addressing a Cluster

A Uniform Resource Identifier (or URI) is a string of characters that identifies a resource over the network. The character string can be organized in a logical format (such as an object name) or a system-generated set of characters organized in a random pattern (such as a UUID):

```
http://companyname.example.com/12BFEA648C2697A56FD5618CAE15D5CA
```

You must know the object's name or UUID to identify the object to any storage cluster that holds at least one replica of the object.

To retrieve an object over a network, you address the cluster and specify a protocol that will deliver the data. A URI as defined in RFC 2396 lets you specify these components in a compact form:

- Protocol (http)
- Cluster name (companyname.example.com)
- Object name or UUID (12BFEA648C2697A56FD5618CAE15D5CA)

Supported Application Protocols

Swarm only understands the application-level protocol called SCSP, which is a subset of the Hypertext Transfer Protocol (HTTP/1.1) used by web servers and browsers. While additional access protocols might be added to Swarm in the future, HTTP is the only protocol supported direct to Swarm at this time.

> For how to access Swarm via the S3 protocol, see S3 Protocol Interface.

Addressing a Cluster

You can address a storage cluster using either:

- The DNS name
- The IP address of a cluster node

Which node you select is not important, as long as the node is accessible to the application on the network. The node named in the URI is not required to be the same node that initially stored the object because any node can be asked to retrieve any object stored on any node in the cluster.

> Important
> If the addressed node is down or off-line, you may need to change the URI you use.

Content Cache

Swarm uses a content cache to store frequently accessed objects (primarily domains and buckets). These objects can be cached in RAM on the requested nodes, which boosts read throughput and response times for relatively small objects that are accessed frequently.

> Important
> To maintain performance, do not disable the content cache unless advised by your support representative, especially if you are writing named objects.

As object demand changes over time, Swarm automatically manages the cache to increase or decrease the number of cached copies throughout the cluster. For most objects, this prevents stale data being returned in a query.

These cluster-wide parameters configure the cache:

- cache.expirationTime
- cache.maxCacheableSize
- cache.percentage
- cache.realmStaleTimeout

> See the Settings Reference for configuring the content cache.
> See Use the Content Cache in a Distributed System.

Response headers for the content cache

The following response headers provide information about the content cache:

- Age. Indicates the length of time (in seconds) the object was stored in the content cache.
  - If the Age header is absent, the object was retrieved from the drive.
  - If the Age header = 1, the object is cached on a node where it also resides on the drive. See RFC 7234 5.1 .

- Cache-Control: no-cache. Matches exactly what was sent with the object on WRITE.
- Cache-Control: max-age. Matches exactly what was sent with the object on WRITE.
- Cache-Control: no-cache-context. Matches exactly what was sent with the object on WRITE.

> See Caching Metadata Headers.

Data Protection

Swarm is designed to provide both fast and secure data storage. Selecting the appropriate trade-offs between performance, capacity, and data protection that best meet your business needs is a critical part of a successful storage solution.

Role of the Health Processor

Swarm includes the Health Processor (HP) that provides end-to-end, disk-level, and lifecycle data protection. The Health Processor monitors both data integrity and cardinality (the number of object replicas) continuously, and it heals any degradation or non-conformity within the storage cluster that it finds.

The Health Processor regularly scans every object on disk to verify its integrity. If the object is corrupt (such as due to a bad sector), HP removes it. HP then detects and corrects the object's under-replication by triggering creation of another replica of that object elsewhere in the cluster, thus restoring the correct number of replicas.

Using the Health Processor, Swarm can:

- Verify the correct object protection level exists in the cluster.
- Distribute replicas and erasure coding segments properly across subclusters.
- Enforce lifepoint policies by enforcing replica and/or erasure coding segment counts and delete policies (specifically, terminal lifepoints) at different policy time intervals.
- Validate the object on-disk integrity and create a new replica if the integrity is compromised.
- Economically store and load balance by periodicallyport evaluating if an object is optimally stored in its current location.
- Defragment storage space on an as-needed basis.

> See Lifepoint Metadata Headers.

Content-MD5 Integrity Checking

In addition to the protection provided by the Health Processor, Swarm uses the Content-MD5 metadata header to provide end-to-end message integrity check of the object body (excluding metadata) as it is sent to and from Swarm.

> See Content-MD5 Checksums.

Understanding Feeds

Purpose of Feeds

Feeds is the object-routing mechanism in the Swarm storage cluster that uses intermittent channel connections to distribute data to:

- Target storage cluster(s) for object replication
- Elasticsearch server(s) for object metadata search

Feeds can replicate all content on the source cluster or all content within a particular domain to a targeted cluster.

The source cluster processes all UUIDs and names stored in the source cluster based on your feed configurations in the Swarm Admin Console Settings page. As objects are added to the cluster, Swarm adds the UUIDs and names to the assigned feed queue. Swarm logic processes the queue and notifies the target cluster and the Elasticsearch server that feeds are available.

All feed changes can take up to 60 seconds to propagate from the source node to the targeted nodes in your cluster.

> Note
>
> A single object can be associated with up to eight feed definitions.

Feeds provide a backup solution for environments with a reliable network connection between the source and target cluster, as well as the source cluster and the Elasticsearch server. In these environments, feeds operate continuously to keep up with source cluster intake.

After an object is replicated, it is not replicated again unless you update a feed. When this occurs, Swarm reevaluates all objects in the source cluster against the new feed definition. If required, Swarm reinitiates another replication to the targeted cluster.

> Tip
>
> To determine whether a particular object was replicated or indexed for metadata search, use administrator credentials to submit an SCSP INFO request for the object and view feed status information in the metadata for the object.

Feed Plug-ins

Swarm supports two types of feed plug-ins that you can implement:

- Remote Replication enables object replication directly to an external storage cluster without API intervention. When Swarm recognizes new or updated objects, it copies these objects to an internal queue. At specific intervals based on the retryWait attribute settings, the plug-in moves the queued objects to the targeted cluster.

  > See Managing Replication Feeds.

- Metadata Search provides real-time metadata indexing and ad-hoc search capabilities within Swarm by name or metadata. The Elasticsearch service collects the metadata for each object and updates the search database in your Swarm network. When you create a new object, domain, or bucket, the service collects only the metadata and not the actual content.

  > See Managing Search Feeds.

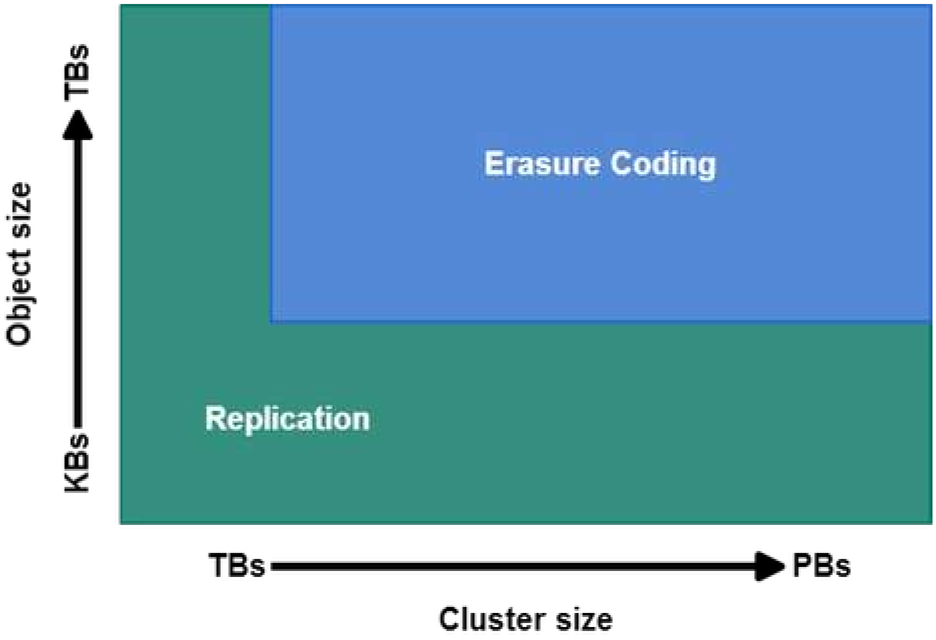
To create feeds on the source cluster that process all UUIDs or names stored in the source cluster or domain, use the Settings page in the Swarm Admin Console.

Elastic Content Protection

Swarm allows you to flexibly determine the type and level of content protection that best fits your storage needs using Elastic Content Protection. Objects can be either replicated or erasure-coded, with objects of both types co-existing in the same cluster.

> Tip
>
> Erasure coding is best suited for clusters with many nodes and larger objects, while replication is

advantageous in smaller clusters and with smaller objects.



Content Protection with Replication

- How Replication Affects Risk
- Controlling Replication Protection
- Increasing Replication Priority
- Enforcing Replicate On Write (ROW)

Swarm can provide protection on disk by creating multiple copies of each object on different nodes called replicas. You can control how many replicas are created for each object and how quickly they are created after the object is initially stored in the cluster.

Note
If one object replica exists in a cluster, there is only one instance of that object in the cluster. In this context, re plica, instance, and object are all synonymous.

How Replication Affects Risk

By default, each object in Swarm is stored with two replicas, with each replica residing on a different node in the cluster. If your cluster is configured to use subclusters, replicas are distributed across subclusters.

In the event of a total failure or a hard drive fails for any reason, the cluster reacts quickly and initiates a volume recovery process for each missing drive. The recovery process rapidly creates additional replicas elsewhere in the cluster of all objects that were stored on the now missing drive(s) so that each object again has two replicas. If a second drive fails before the recovery process is completed, there is a protection risk for the only replica of the object in the cluster.

There can also be a potential period of vulnerability at the moment an object is first stored on Swarm if you do not use the Replicate On Write option to create multiple simultaneous replicas.

Controlling Replication Protection

While a rapid sequence of drive failures is unlikely, it is possible. If this presents an unacceptable risk for your application, the solution is to change your replication requirements. Changing the default replication requirements to a larger number of replicas lets you trade disk space savings for added security.

To set the replication protection for the cluster, you configure a single setting, **`policy.replicas`**, with three required parameters, for **`min`**, **`max`**, and **`default`** number of replicas:

```
policy.replicas: min=2 max=5 default=3
```

Deprecated
The cluster setting policy.replicas replaces the following three, which are all deprecated: **`scsp.minReplicas`** **`scsp.maxReplicas`** **`scsp.defaultReplicas`**

> See Implementing Replication Policy.

Increasing Replication Priority

By default, Swarm writes a new object to one node, responds to the application with a success code and UUID (or name), and then quickly replicates the object as needed to other nodes or subclusters. The replication step is performed as a lower priority task.

While this creates the best balance of throughput and fault tolerance in most circumstances, there are cases where you might want to give the replication task the same priority as reads and writes, which ensures replication occurs quickly even under heavy sustained loads.

Your cluster administrator can add the following parameter to the node or cluster configuration file:

```
health.replicationPriority = 1
```

With replication set to priority 1, object replication is interleaved in parallel with other operations. This might have a negative impact on cluster throughput for use cases involving sustained, heavy writes. With `health.replicationPriority = 1`, it is still possible (though much less likely) that the failure of a node or volume could cause some recently written objects to be lost if the failure occurs immediately after a write operation but before replication to another node can be completed.

Enforcing Replicate On Write (ROW)

Another replication strategy to protect your content is Replicate on Write (ROW).

Without ROW, the client writes a single copy and depends on the Health Processor (HP) to create the necessary replicas. Relying on HP leaves open a small window for data loss: the volume containing the node that holds the only copy could fail before HP completes replication. ROW eliminates that window by guaranteeing that all replicas are written on the initial request.

How it works: The ROW feature requires Swarm to create replicas in parallel before it returns a success response to the client. ROW protection applies to WRITE, UPDATE, COPY, and APPEND requests. When ROW is enabled, the secondary access node (SAN) sets up connections to the number of available peers required to create the needed replicas.

> See Configuring Replicate On Write.

- Implementing Replication Policy

Content Protection with Erasure Coding

Replication is a proven and valuable mechanism to ensure data integrity, but the cost per GB of storage can get high as object sizes and cluster sizes grow. A complementary data protection strategy, erasure coding (EC), provides high data durability with a smaller footprint. Swarm manages EC and replication together to optimize cost-effectiveness, converting objects between them seamlessly and dynamically, based on the policies that you set.

- How EC works
- How much EC protects
- How much drive space EC saves

How EC works

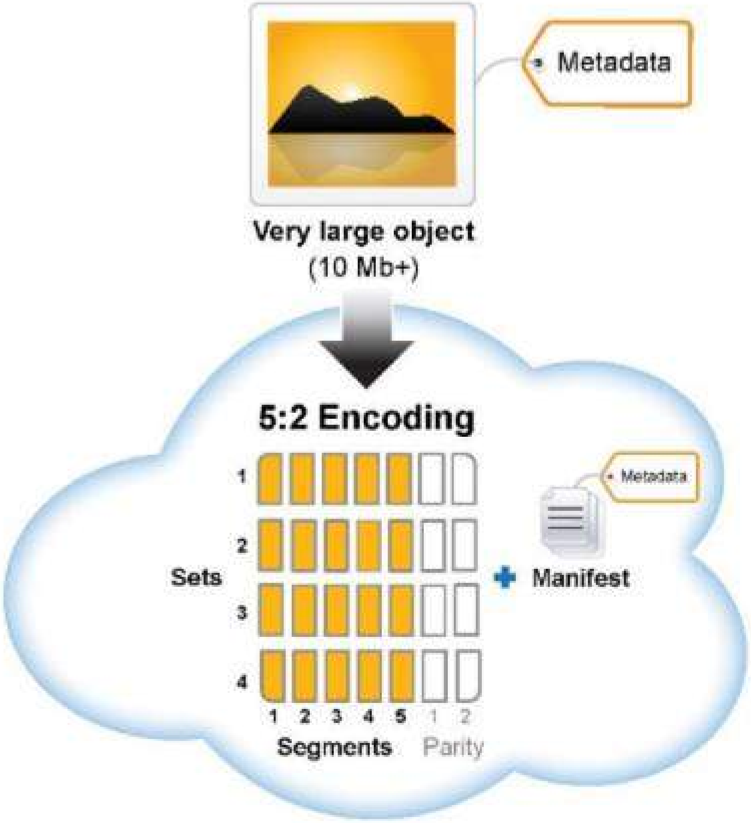
Erasure coding breaks the original object into multiple data segments (k) and computes additional parity segments (p) based on the content of the data segments. This results in m total segments (k + p = m) being distributed to m different nodes (or subclusters) in the storage cluster (see the ec.protectionLevel setting).

The erasure coding encoding level is expressed as a tuple in this format:

```
{data segments}:{parity segments}
```

For very large objects, Swarm creates multiple sets of erasure segments. The object breakdown into one or more erasure sets is transparent to external applications. A GET or HEAD of an erasure-coded object uses the same syntax as a replicated object.

The following illustration represents how erasure coding works:



How much EC protects

If a hard drive or a node containing an erasure segment fails, Swarm can still read the object as long as there are still k total segments (any combination of original data or parity) remaining in the cluster. In other words, the protection against drive failure for the object is equal to the number of specified parity p segments.

For example, because the segments from a 5:2 (5 data segments with 2 parity segments for a total of 7 segments) or 8 :2 (8 data and 2 parity segments for 10 total segments) erasure code are distributed to different nodes, they are protected against the loss of any two nodes. An erasure-coded object is immediately retrievable when accessed even if some segments are missing. However, regenerating the missing erasure set segments is still performed in a self-healing, cluster-initiated manner (similar to the recovery process for replicated objects) to protect against further drive loss. This process kicks off automatically when a missing volume is detected and automatically regenerates any missing segments.

How much drive space EC saves

The amount of drive space (or footprint) used for erasure-coded objects depends on the ratio of data to parity segments in the specified encoding.

Use the following formula to roughly calculate the drive space that you can expect to see used by an EC object with one set of erasure segments:

(total segments ÷ data segments) × object size	= object footprint
((k+p) ÷ k) × GB	= total GB

How footprint changes with different EC encoding (versus 3 reps)

- 1 GB object with 5:2 encoding: ((5 + 2) ÷ 5) × 1 GB = 1.4 GB (vs. 3 GB for replication)
- 3 GB object with 5:2 encoding: ((5 + 2) ÷ 5) × 3 GB = 4.2 GB (vs. 9 GB for replication)
- 3 GB object with 7:3 encoding: ((7 + 3) ÷ 7) × 3 GB = 4.3 GB (vs. 9 GB for replication)

Note
Additional system metadata is written with each EC segment, which adds about 16 bytes per segment.

> See Hardware Requirements for Storage for how to size and optimize your cluster hardware for erasure coding.

- Implementing EC Encoding Policy
- Methods Affected by Erasure Coding
- Conversion between Content Protection Types
- Troubleshooting Erasure Coding

Working with Large Objects

- Dividing Objects with Erasure Coding
- Storing Large Objects
- Storing Streaming Media

To work with very large objects or objects of unknown length, you need to use the advanced options that are incorporated in Swarm Elastic Content Protection:

- Erasure coding (EC), which segments and stores large objects efficiently and securely
- Multipart Write, which divides an object into multiple parts and uploads them simultaneously

These are key terms used in Swarm elastic content protection:

Chunked transfer encoding	Used in WRITE, UPDATE, and APPEND SCSP methods to send objects of an undetermined content length to a storage cluster. The exact request header is: `Transfer-Encoding: chunked.` See RFC 7230 3.3.1. Note COPY rewrites the object manifest only.

Erasure coding	Describes one of the ways an object can be protected in a storage cluster. A large object written to the cluster using erasure coding is automatically stored on disk as a set of data and parity segments. This process ensures both content protection and optimal storage usage for large objects. Swarm has configuration parameters that enable an object to be automatically erasure-coded on the drive.
Manifest	Swarm object containing a list of the segments that comprise a large object.

Dividing Objects with Erasure Coding

Swarm lets you write large objects of known length using the erasure coding option incorporated in the Swarm Elastic Content Protection. With this option, you can divide the object into smaller segments and encode it with additional parity segments that provide data protection.

Additionally, you can write (POST, PUT, COPY, APPEND) objects of unknown length to a cluster using standard HTTP chunked transfer encoding. Objects sent to the cluster using chunked transfer encoding are erasure-coded when stored on disk, using the encoding type specified by either cluster configuration or request query arguments. This feature allows you to store large objects and streaming media in the cluster.

Storing Large Objects

You can store an object as large as 4TB in the cluster. Erasure coding is seamless and transparent to the application, automatically partitioning the object into segments, encoding them, and distributing the segments throughout the cluster. When you configure the cluster, you set the threshold for when objects become erasure coded; in addition, applications can control which objects get erasure-coded on an individual object basis. See Erasure Coding EC.

Attempting to store an object larger than 4TB will result in a 400 Bad Request response immediately after the write is submitted.

> Increasing allowed object size
>
> To store objects larger than 4TB, increase the limit that is set by ec.maxSupported (defaults to 4398046511104) and also set ec.segmentSize (defaults to 200000000) to a value proportionately larger. On a full read, Swarm must load the entire manifest; increasing the segment size minimizes the size of the manifest and so the number of socket connections required to read an entire EC object. (SWAR-7823)

Storing Streaming Media

Streaming media is supported using industry-standard chunked transfer encoding. Your application can now stream digital media or other types of data to the cluster without knowing the object size in advance. The size of the object is limited only by the available space in the cluster (up to 4TB). Attempting to store a chunked encoded object larger than 4TB will result in a 400 Bad Request response (see note above).

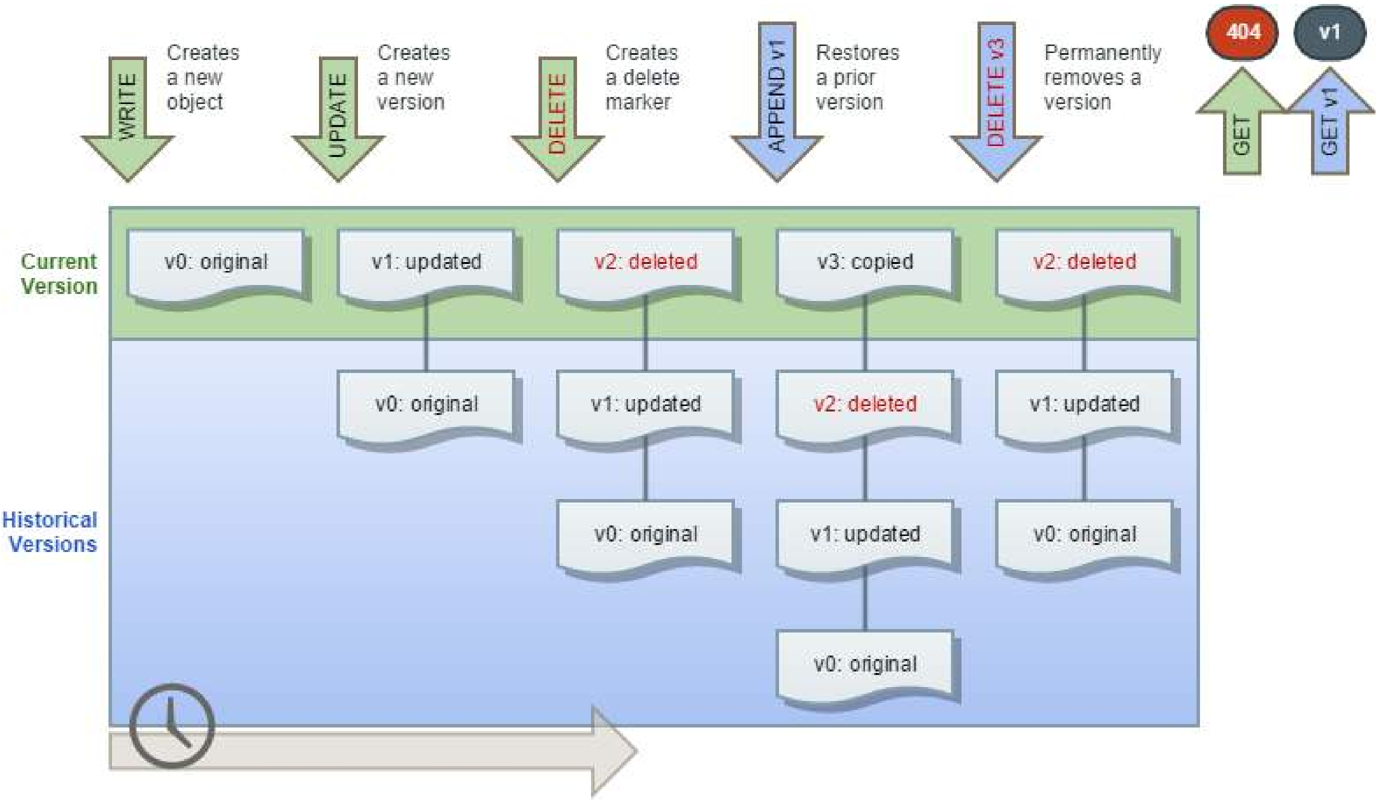
Any object written with HTTP chunked transfer encoding must be erasure-coded and cannot be replicated. If you write an object and specify both erasure coding and replication in the header (for example, combining an encoding=5:2 query argument with a lifepoint header with a reps= parameter), the write operation will result in a 400 Bad Request response.

Version Control for Objects

Object-level versioning is a powerful content protection option that tracks, secures, and provides access to historical versions of objects, even after they are deleted. With versioning, your applications can read, list, revert, and purge prior versions as well as restore objects that were deleted by mistake.

> Note
> Using Swarm versioning with SCSP operations has no dependencies. To use Swarm versioning with Amazon S3, you must run Content Gateway version 4.1 or higher.

Versioning preserves a set of historical variants of an object, the original plus subsequent updates to it, up to and including the latest version:



These are key capabilities of Swarm versioning:

- Unlimited versions – The number of supported versions for a given object is unbounded, and all versions have a unique version ID. You can list all versions, and you can access, restore, and permanently delete specific versions via the version ID.
- Flexible policy – The cluster administrator changes the cluster policy settings to allow versioning; the domain administrator can then allow and even require versioning in that domain. If allowed by the cluster and domain, a bucket owner can enable/disable versioning for a specific bucket.
- Lossless concurrent updates – Swarm captures simultaneous PUT updates and resolves the order in the version chain. Swarm preserves all versions, even those overlapping in time, with the latest update as the current version.

- Accurate disk reporting – Each object revision in a domain/bucket with versioning-enabled preserves and reports its full size on disk. Swarm includes all object revisions in its 'du' responses, if requested, which means the size for deleted and historical versions counts towards bucket and domain totals.
- Support for search and replication – Swarm Versioning works with both Search feedsand Replication feeds, provided that all clusters are running the same version of Swarm.

> Content Router
>
> Content Router is incompatible with Swarm versioning and has been removed from version 8.0.

S3 Versioning

Swarm's native object versioning feature is interoperable with AWS S3 versioning. The implementation includes these improvements:

- Ability to disable versioning:
  AWS S3 only allows for versioning to be suspended once enabled on a bucket. Swarm provides the ability to disable versioning and automatically clean up the prior versions in order to reclaim storage space.
- Delete marker consolidation:
  Unlike AWS S3 where continued DELETE operations on a deleted object will record additional delete markers in the version history, Swarm will acknowledge the subsequent deletes without recording additional delete markers. Multi-factor authentication delete is not supported.
- Expanded version listing:
  Swarm supports version listing batches up to 2000 items while AWS S3 limits these listing results to batches of 1000. Additionally, Swarm does not break batches on version boundaries. Delimiter case is currently not supported for version listing.
- Simplified ACL management:
  When using per-object ACLs with versioning, the ACL for the current version of the object applies for determining authorization. To change the ACL for an object's entire version chain, update the object without spe cifying a version.

Why use versioning?

Versioning meets two key needs:

- You need extremely durable data retention and archiving.
- You need to be able to recover when data is erroneously overwritten or deleted.

With versioning enabled, you can retrieve and restore any of the prior versions of a stored object, which lets you recover from data loss, whether caused by user error or application failure:

- Deleting an object – Instead of removing it permanently, Swarm inserts a delete marker, which becomes the current object version. You can still restore any previous version.
- Overwriting an object – Swarm performs the update by creating a new version, which means that you can roll back a bad update by restoring the previous version.

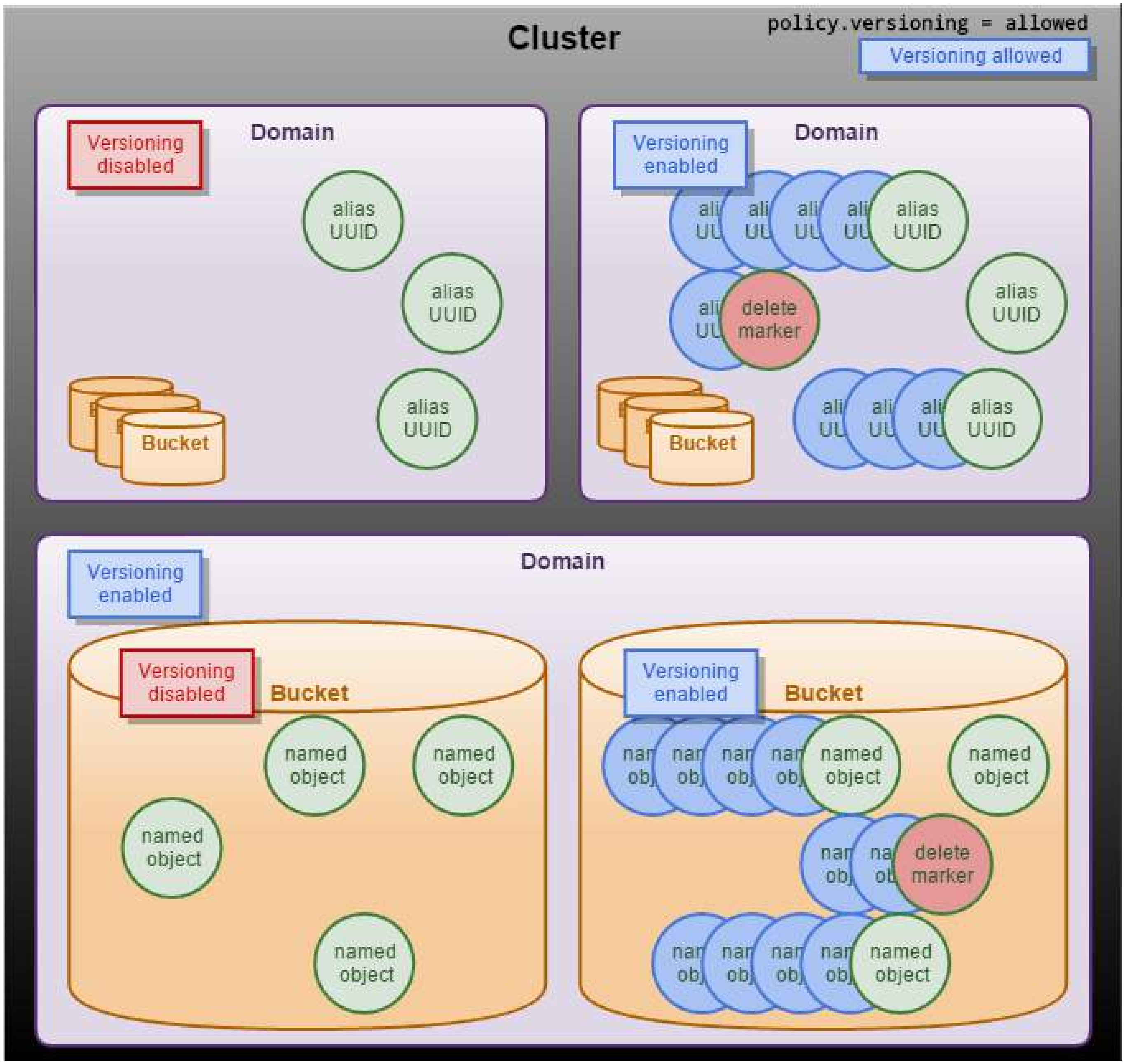
By default, versioning is disabled across the cluster. To avoid excessive storage usage, enable versioning in a targeted way, only where change control is required.

What gets versioned

Choosing to use versioning means that you get the ability to preserve, retrieve, and restore every update of every object that you store in that context (domain or bucket). With Versioning, Swarm archives another copy of an existing object whenever you update or delete it. GET requests retrieve the most recently written version, but you can get back the older versions of an object by specifying a version in the request.

Once you configure the cluster to allow versioning, administrators can selectively enable versioning at

- the domain level (for alias objects)
- the bucket level (for named objects)



When you DELETE a versioned object, Swarm creates a delete marker so that subsequent simple (unversioned) requests will no longer retrieve the object. However, Swarm still stores all versions of that object, so you can retrieve and restore it, if need be.

Note that these types of Swarm objects cannot be versioned:

- Domains
- Buckets
- Unnamed objects (which are immutable)
- Alias objects not tenanted in a domain

Only objects are versioned, not domains or buckets (contexts). This means that, if you accidentally delete a bucket, the bucket is lost; however, Swarm pauses the recursive delete of the bucket's contents for the duration of your grace period (`health.recursiveDeleteDelay`). You have time to recreate the bucket with the same headers and so not lose any data (see Restoring Domains and Buckets). If you choose not to restore the bucket and the grace period expires, the content will start to disappear as Swarm's HP begins cleaning up all versions of the obsolete content, to reclaim space.

- Implementing Versioning
- Working with Versioning
- Versioning Operations
- Versioning Examples

Swarm Metrics

Although you have full access to instantaneous metrics on your storage cluster through SNMP, that route requires you to manage the sampling, recording, and querying of your historical data. The Swarm service for historical metrics, added in 8.1, gives you an easier way to collect the operational metrics and historical time-series data that are so valuable for your administration, billing, and planning activities.

The Swarm Historical Metrics Service uses an Elasticsearch cluster as a data repository, and it samples and records Swarm metrics autonomously at set intervals. You can access these metrics by querying Elasticsearch with aggregating (faceted) queries.

> Tip
>
> You can centralize your metrics management: Multiple Swarm clusters can use the same Elasticsearch instance to collect their metrics.

> See Installing Swarm Metrics.

Components of Historical Metrics

Metrics Curator - The Metrics Curator is the service that does the work of creating and managing your metrics data. It deletes old metrics data, rotates the aliases, and creates new indices. You run the Metrics Curator manually only once, after installation and configuration (to prime the aliases); it is automatic after that, running daily at midnight (GMT) as a `cron` job.

> Note
>
> Only one Metrics Curator service should be running, so install it on one of your Elasticsearch nodes or another server running RHEL/CentOS 7.

Metrics Data - Swarm generates these metrics of interest:

1. feeds

2. index
3. node
4. volume (which includes usage statistics)
5. health
6. memory
7. scsp

These metrics are captured into one Elasticsearch index per cluster, per metric, per day (which means there are 7 × 7, or 49, indexes), with this naming pattern:

```
metrics-{cluster}-{metric}-{date}
metrics-clusterx-health-2018.03.18
```

Aliases combine multiple days of indices together into these useful collections:

```
metrics-{cluster}-{metric}-{alias}
metrics-clusterx-health-all
metrics-clusterx-health-today
metrics-clusterx-health-yesterday
metrics-clusterx-health-this_week
metrics-clusterx-health-last_week
metrics-clusterx-health-this_month
metrics-clusterx-health-last_month
```

Metrics Templates - To support historical metrics reporting, Swarm provides a custom set of Elasticsearch template schemas, which are located here:

`/usr/share/caringo-elasticsearch-metrics/bin`

These template files define the schema of the historical metrics from Swarm:

```
feedsschema.py
healthschema.py
indexschema.py
memoryschema.py
nodeschema.py
scspschema.py
volumeschema.py
```

- Resetting Swarm Metrics
- Swarm Metrics Troubleshooting

Defining Swarm Admins and Users

- Granting Swarm access
- Modifying administrators without rebooting

Granting Swarm access

As of 10.0, Swarm uses two pairs of security lists to grant access to storage cluster management and viewing:

- Administrators can access the Swarm UI and change the cluster configuration. SNMP read/write access is handled separately.
- Operators can only view the Swarm UI. SNMP read-only access is handled separately.

Each user list is specified by a configuration parameter with name value pairs in this format, with the needed passwords for SNMP access being handled in separate settings (v10.0):

```
security.administrators =
{'admin':'adminpassword','admin2':'adminpassword2'}
security.operators =
{'operator':'operatorpassword','operator2':'operatorpassword2'}
snmp.roCommunity = public
snmp.rwCommunity = ourpwdofchoicehere
```

Section notation

```
[security]
administrators = {'admin':'adminpassword','admin2':'adminpassword2'}
operators =
{'operator':'operatorpassword','operator2':'operatorpassword2'}

[snmp]
roCommunity = public
rwCommunity = ourpwdofchoicehere
```

Best practices

The name admin is reserved, so do not delete it, which could cause errors and affect performance. If you decide not to use admin, define a complex password to protect it.

Change passwords from the defaults before putting the cluster into production, and improve security by encrypting the Swarm passwords. See Swarm Passwords.

Modifying administrators without rebooting

You may modify the list of Administrators and their passwords without rebooting by using several read-write SNMP OIDs. New administrative users can be added and existing users modified with the addModifyAdministrator SNMP OID.

- To add a new user, include the new user name and password separated by a colon:

```
addModifyAdministrator = "Jo.Jones:password1"
```

- To modify the password for an existing user, include the existing user name and new password separated by a colon:

```
addModifyAdministrator = "Jo.Jones:password2"
```

- To delete administrative users (except the default admin and snmp users), send the name of an admin user to the removeAdministrator SNMP OID:

```
removeAdministrator = "Jo.Jones"
```

It can take several minutes for these SNMP changes to propagate in the cluster. During this update window, old passwords and deleted users will continue to work for up to 10 minutes.

> Note
> Any changes made via SNMP against a running cluster must also be made in the node or cluster configuration file so that any nodes that are offline when the change is made or new nodes added to the cluster after the fact can correctly authenticate cluster-wide actions.

> Caution
> All administrative users and passwords must agree across all nodes or certain cluster actions will fail.

Managing Domains

This section provides information about domains and describes how to manage them in your Swarm storage cluster.

Domains are secure domains that live entirely within a Swarm storage cluster. Like a storage facility containing multiple storage units, a domain contain multiple buckets that allow you to store unstructured data objects into specific categories, such as documents, photos, and videos.

- Guidelines for Managing Domains
- Manually Creating and Renaming Domains
- Renaming Domains and Buckets

- Accessing Inaccessible Objects with CID
- Recreating Buckets
- Restoring Domains and Buckets
- Resolving Duplicate Domain Names

Guidelines for Managing Domains

When you create domains, follow these guidelines:

- Set up a default cluster domain (a domain name that exactly matches the name of the cluster). Every object that has no domain explicitly defined for it belongs to the default cluster domain.
- Create at least one domain for named objects.
- Ensure that all domain names are unique among all clusters that you manage. If you use an SCSP operation to create a domain with the same name in multiple storage clusters, it creates different domains that share the same name. This leads to name collision and incorrect results if the different domains are replicated into the same cluster. When you create a new domain, create the domain name once and only use Swarm remote replication to copy it into separate clusters.
- Ensure that all domain names are IANA-compliant (for example, `cluster.example.com`). If you currently have a cluster name that is not IANA-compliant, create an IANA-compliant domain name and then create of all your named objects in buckets in that domain. See Naming Rules.

Manually Creating and Renaming Domains

- SDK for Creating a Domain
- cURL for Creating a Domain or Bucket

You can create new domains in your storage cluster any of three ways:

1. Swarm Content UI (preferred); see Configuring Domains
2. Programmatically, using Swarm SDK
3. Manually, using cURL

Best practices

- Use the Content UI to create each new context (domain or bucket), because the UI automatically creates the corresponding domain managers for you and adds the correct protection settings to your cluster. If you need a new domain, contact your cluster administrator to access the Content UI and create a new context for you.
- Do not attempt to create domains and buckets manually unless your administrator is unavailable, you lack Content UI access, and you are an advanced user with experience in creating domains and buckets.
- Create Content Policies (protection settings) for your domains and buckets that match the settings that are normally created by the Content UI. Contact Support for help manually enabling the protection settings.
- Check the Naming Rules before creating any context objects. Unlike cluster names, domain and bucket names cannot include spaces.

SDK for Creating a Domain

The Swarm SDK version 1.4 and later includes classes that can assist you with creating a domain. The following table lists the classes and corresponding source code location in the SDK distribution.

> See the SDK Overview.

Class	Location
C++	`sdk-extract-dir/cpp/src/realm`
C#	`sdk-extract-dir\csharp\ScspCSExamples\ScspRealmExamples.cs`
Java	`CAStorSDK-src-extract-dir/com/caringo/realm`
Python	`castorsdk-python-egg-extract-dir/castorsdk/realm`

cURL for Creating a Domain or Bucket

The example below shows how to create a domain with no domain protection setting, allowing any user to POST to the domain. For guidance on authentication, see Content Gateway Authentication.

To create the domain manually, use the following syntax. Because you are executing SCSP methods on a domain, either the domain query argument or a Host header is required, even if the domain you are creating is the default cluster domain (see Types of Container Objects).

Create a domain

```
$ curl -i -X POST --location-trusted --post301 --anyauth --user
'admin:password' --data-binary '' \
  -H 'Content-type: application/castorcontext' \
  -H "Policy-*: {if needed}" \
 'http://{host}/?domain=newdomain.example.com'
```

Create a bucket

```
$ curl -i -X POST --location-trusted --post301 --anyauth --user
'admin:password' --data-binary '' \
  -H 'Content-type: application/castorcontext' \
  -H "Policy-*: {if needed}" \
 'http://{host}/newbucket?domain=mydomain.example.com'
```

Headers

- Content-type: application/castorcontext — specifies that you are creating a domain or bucket, and it is required when sending requests to the Gateway (see Domain and Bucket Creation). It replaces the createdomain query argument, which is deprecated. The Swarm setting scsp.requireExplicitContextCreate protects objects from being created erroneously as contexts (buckets or domains); with this setting enabled, Swarm will not create a context object unless it includes this header. (v9.1)

- Policy-* headers — add for any domain-specific replication, ec-encoding, and versioning requirements.

> See Configuring Content Policies.

On success, Swarm returns a 201 Created response with the result: `New stream created.`

Best practice
Add conditional headers to ensure that you are updating your intended domain or bucket.

- Use **`if-none-match`** to prevent creating a domain or bucket that already exists:

```
$ curl -X PUT
http://{host}/_admin/manage/tenants/_system/domains/testdomain
-H "if-none-match: *"
```

- Use **`if-match`** on the ETag to prevent updating the wrong domain or bucket:

```
$ curl -X PUT
http://{host}/_admin/manage/tenants/_system/domains/testdomain
-H "if-match: \"ETAG\""
```

Renaming Domains and Buckets

You can rename context objects (domains and buckets) in your storage cluster by using the SCSP COPY command with the newname query argument that specifies the new name.

Tip
You can log in to the Content Portal to verify the renaming.

Important
Be sure to copy existing headers that you want to have preserved.

- In particular, look for Policy-* headers and the x-tenant-meta-name header that Gateway uses to group domains under a tenant.
- Add the preserve query argument to the COPY request to ensure that any custom metadata existing on the object is carried over to the copy. (v9.2)

- To overwrite an existing value, include the header name with the new value on the request.

See Headers to preserve in SCSP COPY.

Renaming a Domain

To rename a domain, use this syntax:

```
curl -i --location-trusted -u admin -X COPY
  'http://{host}?domain={old- name}&admin&newname={new- name}&preserve'
 [-D log-file-name]
```

Example

```
curl -i --location-trusted -u admin -X COPY
 'http://172.16.0.35?domain=abc.example.com&admin&newname=xyz.example.co
m&preserve'
 [-D log-file-name]
```

The message New object created confirms that the renaming procedure was successful.

Renaming a Bucket

To rename a bucket, use this syntax:

```
curl -i --location-trusted -u admin -X COPY
 'http://{host}/{old- name}?admin&newname={new- name}&preserve'
 [-D log-file-name]
```

Example

```
curl -i --location-trusted -u admin -X COPY
 'http://172.16.0.35/bucketold?admin&newname=bucketnew&preserve'
 [-D log-file-name]
```

The message New object created confirms that the renaming procedure was successful.

Accessing Inaccessible Objects with CID

If a domain, bucket, or named object is inaccessible by name, you can still access it by ID using a Context Identifier (CID) query argument. The CID query argument syntax is: cid=

This troubleshooting process is helpful when:

- The domain or bucket was deleted.
- The domain was duplicated in a disaster recovery cluster (two domains exist with the same name in the same cluster). (See Resolving Duplicate Domain Names.)

> Note
> This procedure lets you access the object but not to recover it. To recover accidentally deleted domains and buckets, see Restoring Domains and Buckets.

To execute a cid= query argument, you need the value of the object's Castor-System-CID header that identifies the object's parent. For example, if an object named `photo1.jpg` is not accessible, locate the value of its Castor-System-CID header.

If you did not record or store this information, you can locate it one of two ways:

- Review the debug-level system logs. These logs record the Castor-System-CID value every time you access the object.
- Use the Content Router Enumerator. (Swarm 7.2 or prior only) This tool iterates through all objects in a cluster and returns information about each object.
  To implement this tool:
  1. Add a Content Router filter rule to search for objects where the value of the CastorSystem-Name header is the name of the inaccessible object.
  2. Using the SDK, instantiate a metadata enumerator subscribed to the rule channel you created in the preceding bullet to obtain the object's metadata.
  3. In the metadata returned for the object, look for the value of the Castor-System-CID header.

> See the Enumerator setup in the SDK Overview.

After you locate the value of the object's Castor-System-CID header, access the object using the cid=CID-header-value query argument. To access a named object using a web browser, enter the following URL in the browser's address or location field:

```
http://node-ip/object-name?cid=CID-header-value
```

For example, to access an object named file.html with a CID of 55aba17ad53c61782d7dd0afa8dd2f7d, enter:

```
http://node-ip/file.html?cid=55aba17ad53c61782d7dd0afa8dd2f7d
```

Recreating Buckets

You can delete a bucket and recreate another bucket with the same name. Be aware that the new bucket is a different bucket (a different ID) that happens to have the same name.

After you delete a bucket, all objects in that bucket are inaccessible, even if you subsequently create another bucket with the same name.

> Best practice
>
> For best results, wait at least twice the value of the cache.realmStaleTimeout parameter before you recreate a bucket with the same name as a bucket you just deleted.

If the cache.realmStaleTimeout is still at the default value of 10 minutes (600 seconds), delete the bucket, wait 20 minutes, and then create the new bucket.

Restoring Domains and Buckets

- Recovering a Deleted Domain
- Recovering a Deleted Bucket

If deleting a domain or bucket (context object) in your storage cluster was done by mistake, you can recover it by recreating the object using the recreatecid query argument.

> Caution
>
> Any mistakes in using these commands can cause serious problems. Consult Support for assistance with these operations.
>
> Do not attempt to use recreatecid to move bucket contents across domains within the cluster.

> Tip
>
> If you are within the grace period (health.recursiveDeleteDelay) following the recursive deletion of a domain or bucket, you can use the special methods below to restore it without data loss. If the deletion had no grace period (recursive=now), only some of the data may have been lost to reclamation, depending on how much time has passed since the delete.

Recovering a Deleted Domain

To recover a domain, you cannot just create a new one with the identical name, because it will be mapped to a new UUID. Instead, you create a new domain from the command line and use a query argument to apply the previous domain's UUID.

Orphaned buckets - Because the buckets in the deleted domain reference it not by name but by UUID (Castor-System-CID), you cannot access those buckets until you create a new domain that uses the original UUID.

To recover the domain:

1. Locate and record the log message related to the missing domain.

```
Domain 'example.com' (uuid=a2fc4bb0fc31bbc73a088783aef8ea73) has
been deleted ...
```

2. Copy the UUID listed within the log message, which you need to recreate the missing domain.

```
a2fc4bb0fc31bbc73a088783aef8ea73
```

3. Create a new domain with a POST that references the deleted domain's UUID in the recreatecid query argument:

```
$ curl -i -X POST --location-trusted --post301 --anyauth --user
'admin:password' --data-binary '' \
  -H 'Content-type: application/castorcontext' \
  -H "Policy-*: {if needed}" \

'http://{host}?domain=example.com&admin&recreatecid=a2fc4bb0fc31bbc
73a088783aef8ea73'
```

Recovering a Deleted Bucket

After you delete a bucket, the named objects within the bucket are inaccessible until you recover the bucket.

> Warning
> Do not use `recreatecid` as a way to move bucket contents across domains within the cluster; this will cause critical errors.

To recover the bucket:

1. Locate and record a critical log message related to the missing bucket, resulting from a named object within it being inaccessible.

```
Bucket 'mybucket' (uuid=75edd708dc250137849bbf590458d401) in domain
'example.com' has been deleted with orphan content.
Consider recreating.
```

2. Copy the UUID listed within the log message, which you need to recreate the missing bucket.

```
75edd708dc250137849bbf590458d401
```

3. Create a new bucket with a POST that references the deleted bucket's UUID in the recreatecid query argument:

```
$ curl -i -X POST --location-trusted --post301 --anyauth --user
'admin:password' --data-binary '' \
  -H 'Content-type: application/castorcontext' \
  -H "Policy-*: {if needed}" \
 'http://{host}/mybucket?domain=example.com&admin&recreatecid=75edd
708dc250137849bbf590458d401'
```

Resolving Duplicate Domain Names

- Renaming a Domain in its Source Cluster (DR Cluster Conflict Only)
- Manually Renaming a Domain or Bucket in a Mirrored or DR Cluster
- Example of renaming a domain

When creating domains, you must ensure that all domain names and all bucket names within a particular domain are unique among all clusters you manage. This is particularly important if you are replicating from one cluster to another. Using Swarm Feeds, you can create two types of DR cluster configurations:

- DR Cluster. Copies one or more clusters and their contents in another physical location.
- Mirrored Configuration. Copies the contents of cluster 1 to 2 and the contents of cluster 2 to 1.

In either type of configuration, if two clusters contain two unique domains/buckets with the same external name, Feeds replication will create a duplicate domain/ bucket name(s) in the DR or mirrored cluster. This results in indeterminate access to objects in the duplicated domains/buckets. Sometimes a request to a particular object in one of the duplicate domains/buckets succeeds, but other times it fails.

When Swarm detects a duplicate, it logs a Critical error to its Admin Console similar to the following:

```
SCSP CRITICAL: Domain
'collisiondomain.e0f55af9abcacd625cfd946a1a5e49d0'
(uuid=15748c61aea50ec3bcdd28df763f6cfa) has collided with existing
Domain
(uuid=6ba3aeda10f2254e5b418b73c684c838).
Remove or rename one of the versions to avoid conflict.
```

If you receive a Critical error, perform one of the following procedures:

- Rename a duplicate domain from the Admin Console in the replication source cluster (recommended for a DR cluster conflict). This method resolves the issue and prevents it from happening in the future.

  > Note
  > This method does not work in a mirrored configuration because both clusters have duplicates. In this situation, use the next procedure.

- Manually rename either conflicting domain or bucket in either cluster. This is the only method you can use in a mirrored cluster conflict. It resolves the issue and prevents it from reoccurring. For a DR cluster conflict, this method is not recommended because the next time the same domain or bucket is replicated to the DR cluster from its source, the duplicate domain name still exists.

Renaming a Domain in its Source Cluster (DR Cluster Conflict Only)

This section describes how to rename a domain in its source cluster where the name of the domain is assumed to be unique. After you rename the domain, it replicates without errors to the DR cluster.

For a conflict in a mirrored configuration or if you are using Content Gateway and domain creation via the Swarm Admin Console is disabled, see Manually Renaming a Domain or Bucket in a Mirrored or DR Cluster.

To rename a domain in the source cluster of a DR cluster:

1. Open the Swarm Admin Console, and click Settings.
2. On the Cluster Settings page, click Edit next to the name of the domain you want to rename.
3. In the Add Cluster Domain section, enter a new name in the Domain Name field.
4. Click Save.
5. If prompted, enter an administrator user name and password.

Manually Renaming a Domain or Bucket in a Mirrored or DR Cluster

To manually rename a domain or bucket in a mirrored or DR cluster, use the SCSP COPY command with the following query arguments and authenticate as a cluster administrator.

Query Argument	Meaning
`admin`	Also called as administrative override, this query argument lets you ignore `Allow` headers and bypass the `Castor-Authorization` header. Important: This query argument must be used with cluster admin credentials. Note Administrative override does not affect lifepoint policy deletability for immutable objects.
`newname=new-domain-name`	The new name for the domain or bucket. See Guidelines for Managing Domains.
`aliasuuid=domain-UUID`	The UUID of the domain or bucket to rename. You can find the UUID in the critical log message printed when a duplicate is detected.

To rename a domain in a mirrored or DR cluster:

1. Record the alias UUID for the duplicate domain or bucket you want to rename from the related critical error message.
2. HEAD the alias UUID of the domain or bucket to see if it has protection settings that need to be modified.

Use the SCSP INFO command as follows:

```
INFO /alias- uuid?admin Host: domain- name- or- ip
```

3. You must authenticate as a cluster administrator (that is, a user in the security.administrators parameter). If present in the HEAD results, you must also get the value of the CastorAuthorization header and pass it in using the new cluster name with the rename command as shown in the next step.
4. Rename the domain or bucket.

```
curl -X COPY
 -H 'Castor-Authorization: value- from- HEAD- if- present'
 -H 'lifepoint: [] reps=16'
 -H 'Castor-Stream-Type: admin'
 --anyauth -u 'admin:password'
 --location-trusted
  'http://nodeip?domain=domain-name
   &admin
   &aliasuuid=uuid
   &newname=new- domain- or- bucketname'
```

5. Drop and recreate your search feed in order for the new name to be updated in search and listing requests (See Managing Search Feeds).

Example of renaming a domain

For example, to rename the cluster.example.com domain to archive.example.com by sending commands to a node whose IP address is 172.16.0.35:

1. Record the alias UUID for cluster.example.com from the related critical error message (bbc2365b3283c23c4759 5abcfd09034a in this example).
2. HEAD the domain to get its protection settings, if any:

```
curl -i
 --anyauth -u 'admin:ourpwdofchoicehere'
 --location-trusted 'http://172.16.0.35/alias-uuid?admin'
```

Sample output:

```
HTTP/1.1 200 OK
 Cache-Control: no-cache-context
 Castor-Authorization: cluster.example.com/_administrators,
POST=cluster.example.com
 Castor-Stream-Type: admin
 Castor-System-Alias: bbc2365b3283c23c47595abcfd09034a
 Castor-System-CID: ffffffffffffffffffffffffffffffff
 Castor-System-Cluster: cluster.example.com
 Castor-System-Created: Wed, 17 Nov 2010 15:59:13 GMT
 Castor-System-Name: cluster.example.com
 Castor-System-Owner: admin@CAStor administrator
 Castor-System-Version: 1290009553.775
 Content-Length: 0
 Last-Modified: Wed, 17 Nov 2010 15:59:13 GMT
 lifepoint: [] reps=16
 Etag: "099e2bc25eb8346ed5d94a598fa73bfa"
 Date: Wed, 17 Nov 2010 16:02:07 GMT
 Server: CAStor Cluster/5.0.0
```

3. If a Castor-Authorization header is present, you must update it to reflect the new name for the duplicate. If it is not present, you do not need to add it.
   The information you need is:

   ```
   Castor-Authorization: cluster.example.com/_administrators,
   POST=cluster.example.com
   ```

   You must change this header to:

   ```
   Castor-Authorization: archive.example.com/_administrators,
   POST=archive.example.com
   ```

   You must also add the following headers exactly as shown:

   ```
   -H 'Castor-Stream-Type: admin'
   -H 'lifepoint: [] reps=16'
   ```

   lifepoint: [] reps=16 enables the domain to be replicated as many times as possible.

Use Castor-Stream-Type: admin for all objects that use a Castor-Authorization header.

4. Rename the domain.

```
curl -i -X COPY
 -H 'Castor-Authorization: archive.example.com/_administrators,
POST=archive.example.com'
 -H 'Castor-Stream-Type: admin'
 -H 'lifepoint: [] reps=16'
 --anyauth -u 'admin:ourpwdofchoicehere'
 --location-trusted
  'http://172.16.0.35?domain=cluster.example.com
   &admin
   &aliasuuid=bbc2365b3283c23c47595abcfd09034a
   &newname=archive.example.com'
 -D rename-domain.log
```

5. Drop and recreate your search feed in order for the new name to be updated in search and listing requests. (See Managing Search Feeds.)

Configuring the Nodes

This section describes how to configure the cluster nodes in your Swarm storage cluster.

To configure the cluster parameters, modify the node or cluster configuration in one of the following ways:

- If your cluster boots from a Platform Server, it is already set up to PXE boot.
- If your cluster does not boot from a Platform Server, you can:
  - PXE boot the cluster
  - Boot the cluster using a centralized configuration server
  - Boot from a USB flash drive (requires you to update the caringo/node.cfg file in each node with identical settings)

> Note
> Because of the variety of ways you can configure your cluster, the configuration files are referred to collectively as the cluster or node configuration file.

- Managing Configuration Settings
- Settings Reference
- Configuring Content Policies
- Configuring Content Integrity Settings
- Configuring Volumes Options
- Configuring Power Management
- Configuring the Overlay Index
- Configuring External Logging
- Configuring an Rsyslog Server

Sections. You have the option to organize your cluster settings into sections. A configuration file section is identified by a unique name within square brackets ([section-name]) that contains settings that are logically related to each other.

Guidelines: To manage your configuration files in your storage clusters, follow these guidelines:

1. Set the appropriate configuration settings.
2. If upgrading, add, remove, or adjust the settings and values as directed.
3. Use new and empty sections properly.

Scope of settings

Each setting has one of two scopes:

- cluster-wide: Used across the entire cluster. This is the default scope, so it is not specified in the reference. Every cluster-wide setting must be identical for all nodes in the node or cluster configuration file.

> Warning
> Any difference in the cluster-wide configuration settings among the nodes can cause serious errors.

- node-specific: Specific to each node. These are the exceptions to the majority (cluster-wide), so they are flagged in the reference. Each cluster node can have a different value.

Qualified versus unqualified settings

Swarm supports both qualified and unqualified configuration settings.

- An unqualified setting (such as `corporate`) must be contained in its designated [section], in the empty [] section, or in a configuration file that has no sections (or only the empty section).
- A qualified setting (such as `networks.corporate`) is the name of the section, a period, and the unqualified setting name.

A configuration file can have a mixture of unqualified and fully qualified settings, provided the settings are in the proper sections or the configuration file has no sections (or only the empty section).

Organizing settings by section

In Swarm, sections are optional (unless you use new, unqualified names). However, if you use sections, you must use them properly.

New sections

If you place unrelated parameters or an old parameter name in any section, errors result and Swarm will not start. For example, the following parameter is not valid:

```
[health]                   # Swarm will not start
misplaced-setting = 900
```

The following setting is valid:

```
[health]
startDelay = 900
```

Empty sections

The empty section [] (empty set of square brackets) is a special container that clears any section that might have preceded it. For example, the following setting is valid:

```
[health]
startDelay = 900
[]
ipaddress = 192.168.0.33
```

Formatting settings

All configuration files contain the name-and-value pairs for setting configuration options.

- The format for these pairs is `name = value`.
- Whitespace before the name, between the name and =, and between = and the value is ignored.
- Quotes (`"`) are not stripped from values.
- Blank lines and lines beginning with a hash tag (#) are ignored. However, if this tag appears within the parameter value (anywhere after =), it is part of the option's value field and not a comment.

Important
Settings and values are case-sensitive, and names must have no spaces.

Warning
Spelling or case errors or illegal values can prevent the node or cluster from booting.

Upgrading your configuration file

Update your node or cluster configuration file by placing new settings, renamed settings, and sections at the end of your existing configuration file. This will prevent conflicts with settings located at the beginning of the file.
When you add new settings, use the following guidelines:

- Add new configuration settings in sections or a fully qualified format. Adding new settings at the end of your configuration file avoids problems that can prevent the cluster from booting. For example, to add the new encoding setting, place it at the end of your node or cluster configuration. In this scenario, the new setting does not affect any settings before it. The cluster boots normally and you do not have to change anything else in the configuration file.
- Rename existing settings. Settings that have been renamed are deprecated and will removed eventually in a future release.
  - See the Renamed Settings.
- Add sections to your configuration file, ensuring that all settings are in the correct sections. If you put a section in your node or cluster configuration file, all settings after that section must either be in their respective sections or must be fully qualified.
  - See the Configuration Settings.

Settings Reference

Following are the published settings that let you configure how your storage cluster nodes operate. They are sorted alphabetically by section, and some do not appear in the provided `node.cfg.sample` configuration file. To change these settings, use the Swarm UI.

- Cluster Settings
- Node (Chassis) Settings

> For settings that have changed from earlier versions, see Renamed Settings.

Dynamic settings and SNMP

Many cluster settings (and a few node settings) are dynamic, accepting runtime changes without any hardware restarting. Dynamic settings are those that have SNMP names defined below. In the Swarm UI, settings that cannot be changed dynamically are flagged with a restart icon: the changes you make are applied and take effect after the next restart.

To change settings by SNMP, see Persisted Settings.

Tip

The special value -1 is often used to direct Swarm to take the value of another setting. Follow the guidance in the setting description to use it correctly in each context.

Cluster Settings

Name SNMP Name	Default	Description Examples

bidding.relocationThreshold SNMP: relocationThreshold	5	Percentage, 0-100. How much difference between volume utilizations will cause a lower bid on another node to relocate or rebalance a replica to the other node. Lower values improve load balancing and throughput. Higher values minimize data movement at the expense of lower maximum throughput.
cip.group SNMP: group	224.0.10.100	The multicast IP address for the cluster, as a Class D IP address in the 224.0.0.0 - 239.255.255.255 range. This address must be unique for each cluster. When configuring multiple, distinct clusters, take care that the multicast groups do not overlap, as any node with the same multicast group will become part of a single cluster. Examples: 224.5.5.7 239.255.255.253
cip.queryRetryMultiplier SNMP: queryRetryMultiplier	1	What multiple of time to wait on each successive UDP multicast read retry.
cip.ttl	1	Controls configuration of multicast network traffic TTL (time to live). When set to 1, the multicast traffic should remain on the subnet.
cluster.enforceTenancy SNMP: enforceTenancy	FALSE	Whether to enforce tenancy, which determines where a new unnamed object is created if no domain is specified. If True (recommended), it is tenanted in the default domain; if False, it is written outside of the default domain. Gateway requires enforceTenancy to be enabled, and it is best practice to tenant all objects. This setting exists to preserve backwards compatibility.

cluster.name SNMP: cluster		Optional. The name of the cluster. Use an IANA-compatible domain name, such as cluster.example.com, and create one domair with the same name as the cluster, which sets up a default cluster domain that holds all unnamed objects. Do not use spaces in the name. To prevent confusion, configure all nodes in the cluster with the same cluster name. Example: swarm1.yourcompany.com
cluster.proxyIPAddress		[deprecated] The reverse proxy IP address for the cluster. Use cluster.proxyIPList. Example: 129.3.7.14
cluster.proxyIPList		For use with bidirectional GET replication only, to configure proxies on the source side for the target nodes to connect to. A comma-separated list of reverse proxy IP addresses or names, including ports in name:port format. Example: 129.3.7.14:80, 129.3.7.15:80
cluster.proxyPort	80	[deprecated] The reverse proxy access port for the cluster. Use cluster.proxyIPList
console.messageExpirationSeconds SNMP: messageExpirationSeconds	1209600	In seconds; defaults to 2 weeks. How long until an error expires out of the error table.
disk.atimeEnabled SNMP: accessedTimeEnabled	FALSE	Whether to track the time of last access on GET requests, stored in the Castor-System-Accessed header and indexed as the search field 'accessed'. Increases load on the cluster and Elasticsearch.
disk.atimeGranularity SNMP: accessedTimeGranularity	86400	In seconds; defaults to 1 day. The window during which accessed time will not be updated. Lowering the value affects GET performance.
disk.contextDeleteMarkerLifespan	31536000	In seconds; defaults to 1 year. How long a delete marker lives for a context (domain or bucket) object.

disk.deleteMarkerLifespan	1209600	In seconds; defaults to 2 weeks. How long the cluster remembers a deleted named object. Lower this value if your applications create and delete objects so rapidly that they cause available memory to decrease. To view the current amount of available memory on a node, expand Node Info to see the value of Index Utilization. If this value is high for a long period of time, you may have stored a large number of objects and may benefit from lowering this value.
disk.obsoleteTimeout	1209600	In seconds; defaults to 2 weeks. The amount of time after which an unused volume is considered "stale" and will not recover, except with use of the 'k' modifier.
ec.conversionPercentage SNMP: ecConversionPercentage	0	Percentage, 0-100; 5 is recommended. Adjusts the rate at which the Health Processor consolidates multi-set erasure-coded objects each HP cycle. Setting it to 0 stops all conversion.
ec.maxManifests	6	Range, 3-36. The maximum number of manifests written for an EC object. Usually p+1 are written for a k:p encoding. Do not set above 6 unless directed by Support.
ec.minParity	-1	Range -1 or 1-4; default of -1 is max(policyminreps - 1, 1). The minimum number of parity segments the cluster requires. This is the lower limit on p for EC content protection, regardless of the parity value expressed on individual objects through query arguments or lifepoints. The value 'policyminreps' refers to the min value in policy.replicas.
ec.protectionLevel	node	Either 'node', 'subcluster', or 'volume'. At what level segments must be distributed for an EC write to succeed; note that multiple segments are allowed per level, if needed. 'node' (default) distributes segments across the cluster's physical/virtual machines. 'subcluster' requires node.subcluster to be defined across sets of nodes. You must have (k+p)/p nodes/subclusters for those levels; at minimum, you must have k+p volumes.

ec.segmentConsolidationFrequency SNMP: ecSegmentConsolidationFrequency	0	Percentage, 1-100, 0 to disable; 5 is recommended, which performs all consolidations over 20 HP cycles. How quickly the health processor consolidates object segments needing post-ingest consolidation. Increase this value (such as to 25, which consolidates over 4 HP cycles) to optimize how efficiently newly ingested data is read by clients. Consolidation changes the ETag (which affects If-Match conditional requests) and Castor-System-Version headers, but any Content-MD5 and Composite-Content-MD5 headers are unchanged. Therefore, have clients use the hash and last-modified date, rather than ETag, to determine whether an object has changed.
ec.segmentSize	-1	In bytes; default of -1 implies 200 MB, with recommended minimum of 100 MB. The maximum segment size for each erasure-coded data segment in an erasure set before another erasure set is created.
health.defragInterval SNMP: healthDefragInterval	3600	In seconds; defaults to 1 hour. How long to wait between attempts to defrag a volume during an HP cycle.
health.neonatalROWProtection	TRUE	If the exam queue for newly written objects is close to overflow, enables Swarm to override the data protection scheme of transitioning to ROW (scsp.replicateOnWrite). All subsequent replicas are processed out of this queue.
health.offloadPauseInterval SNMP: healthOffloadPauseInterval	600	The delay between attempts to bulk offload to the cluster, in seconds.
health.persistentUnderreplicationAlertPercent	2	Percentage, 0-100; set 0 to disable. Creates an alert when this percentage (or more) of objects are persistently under-replicated.

health.recursiveDeleteDelay	604800	In seconds; defaults to 1 week. The length of the grace period before the health processor begins reclaiming the space for a deleted domain or bucket. During this grace period, you can restore the domain or bucket without losing any of its content. No grace period is granted if you use recursive=now.
health.relocationVolumeFillRate SNMP: hpRelocationVolumeFillRate	10	Percentage, 0-100. How much available space on new volumes may be filled for object relocation during one cluster health processor (HP) cycle, to prevent the HP on existing nodes from overwhelming a new, empty node.
health.replicationMulticastFrequency SNMP: repMulticastFrequency	1	Percentage, 0-100. The frequency, as an approximate percentage, that UUIDs are multicast to verify replicas. Set this parameter to the same value for all nodes in the cluster.
health.replicationUnicastFrequency SNMP: repUnicastFrequency	100	Percentage, 0-100. The frequency, as an approximate percentage, that a unit is forced to verify hints.
health.underreplicationAlertPercent	10	Percentage, 0-100; set 0 to disable. Generates an under-replication alert when the percentage of under-replicated objects exceeds this value.
health.underreplicationTolerance	100	Count. The number of under-replicated objects below which to suppress the alerts triggered by health.underreplicationAlertPercent.
index.optimize404 SNMP: overlayOptimize404	TRUE	Enables the Optimize 404 feature in the overlay index, which returns 404 without multicast where possible.
index.ovMinNodes SNMP: overlayMinNodes	32	Count. The minimum number of cluster nodes needed to activate use of the overlay index. Lowering the number of nodes may affect throughput performance.
index.overlayEnabled SNMP: overlayIndexEnabled	TRUE	Enables the overlay index.

log.host SNMP: logHost		The IP address of the remote Syslog server. Logging must be used for production environments. Set to '' to stop logging in test environments. Example: 10.10.33.12
log.level SNMP: logLevel	40	The log level, from most to least verbose, each including everything below it: 10, 20, 30, 40, 50, 0. 10 Debug (all information plus stack traces), 15 Audit (replication and object movement), 20 Info (informational, including non-errors), 30 Warn (user and application errors, plus SCSP 4xx/5xx codes), 40 Error (server hardware and software errors, plus abnormal conditions), 50 Critical (errors that can result in data loss, such as disk I/O errors), 0 Disable logging.
log.port SNMP: logPort	514	The port for the remote syslog host to use.
metrics.diskUtilizationCheckInterval	600	In seconds, from 15 seconds to 1 day; defaults to 10 minutes. How frequently to check disk utilization on the Elasticsearch cluster.
metrics.diskUtilizationThreshold	5	Percentage, 0-100. The minimum space available Elasticsearch disk space that, when reached, will stop metrics from being indexed.
metrics.period SNMP: metricsPeriod	900	In seconds, from 15 seconds to 1 day; defaults to 15 minutes. How frequently to capture metrics-related statistics.
metrics.port SNMP: metricsTargetPort	9200	The port on the Elasticsearch server where metrics-related statistics are captured.

metrics.target SNMP: metricsTargetHost		One or more servers in the Elasticsearch cluster (fully qualified domain names or IP addresses) where metrics-related statistics are captured. Use spaces or commas to separate multiple values. To disable statistics collection, leave blank. Examples: es1.yourcompany.com, es2.yourcompany.com 10.12.14.14
network.igmpTimeout SNMP: networkIGMPTimeout	0	In seconds; defaults to 0 (disabled). The IGMP querier timeout, which is the frequency that IGMP queries will be sent on the network.
policy.eCEncoding SNMP: policyECEncoding	unspecified anchored	The cluster-wide setting for the EC (erasure coding) encoding policy. Valid values: unspecified, disabled, k:p (a tuple such as 5:2 that specifies the data (k) and parity (p) encoding to use). Add 'anchored' to prevent overrides of the policy within the cluster. Examples: 5:2 6:3 anchored
policy.eCMinStreamSize SNMP: policyECMinStreamSize	1Mb anchored	In integer units of megabytes (MB) or gigabytes (GB); defaults to 1MB. The size that will trigger an object to be erasure-coded, if specified (by eCEncoding, lifepoint, query arg) and allowed by policy. Objects below this threshold are fully replicated. Examples: 100Mb 1GB anchored
policy.replicas SNMP: policyReplicas	min:2 max:16 default:2 anchored	The min, max, and default replicas allowed for streams in this cluster Examples: min:2 max:16 default:3 min:3 max:10 default:3

policy.versioning SNMP: policyVersioning	disallowed	Specifies whether versioning is allowed within the cluster; by default, it is disallowed. Valid states: disallowed, suspended, allowed. This policy overrides context-level policies. Examples: allowed disallowed suspended
power.savingMode SNMP: powerSavingMode	TRUE	Enables Power Saving mode, which allows the system to go to sleep or power cap. Set to False to disable Power Saving mode.
power.sleepAfter SNMP: sleepAfter	7200	In seconds, 60 or greater; defaults to 2 hours. In Power Saving mode, how long a node is inactive before it becomes idle.
power.wakeAfter SNMP: wakeAfter	28800	In seconds; defaults to 8 hours. In Power Saving mode, how long a node is idle before it becomes active again.
recovery.completedRecoveryExpiration SNMP: completedRecoveryExpiration	2592000	In seconds; defaults to 30 days. How long to remember completed recoveries.
recovery.suspend SNMP: volumeRecoverySuspend	FALSE	Defaults to False, which allows normal volume recovery and recovery behavior. Set to True to disable all recovery behavior. All nodes in the cluster must be set to the same value.
recovery.suspendedVolumes	[]	The comma-separated list of 32-character volume IDs of the volumes for which recovery is suspended. Example: ['d315ca82bae4b4a0d24fd90904216554', '2195a057c205bd58e05f5835d4b9f21e']

recovery.volMaintenanceInterval SNMP: volMaintenanceInterval	10800	In seconds; defaults to 3 hours. How long the cluster waits after a node has been rebooted or shut down before considering the node and its volumes missing for recovery and replication purposes. This time does not include the time to mount the volumes. This maintenance window allows administrators to perform regular, scheduled tasks on a node without creating over-replication in the cluster. Node shutdowns or failures that are not initiated by an administrator are considered immediately missing.
scsp.allowPutCreate SNMP: allowPutCreate	FALSE	When true, PUTs can be used to create new named objects. Conditional headers still apply. With this option enabled, you do not need to add the putcreate query argument.
scsp.autoContentMD5Computation SNMP: autoContentMD5Computation	FALSE	When true, Swarm computes and stores the Content-MD5 value on every applicable write
scsp.autoRecursiveDelete SNMP: autoRecursiveDelete	TRUE	When true, all context deletes (deletes of domains and buckets) are treated as recursive, which prevents orphaned content. With this option enabled, you do not need to add the recursive query argument. To force immediate reclamation of space, use the recursive=now argument.
scsp.clientPoolTimeout	120	In seconds. How long until pooled SCSP connections expire.
scsp.defaultContextReplicas	-1	Defaults to -1, which uses the value of scsp.maxContextReplicas. Sets the default number of replicas for a POST/PUT on a context (domain or bucket) object if the number is not specified by the current lifepoint or the request.
scsp.defaultROWAction	immediate	The default Replicate On Write (ROW) action when scsp.replicateOnWrite is enabled. Valid options are 'immediate', 'full', or an integer between 2 and 5 (inclusive).

scsp.domainHeaders	['X-Forwarded-Host', 'Host']	A comma-separated list of headers that specifies the search order in which to find the host of an SCSP request. RFC 7230 5.4 requires a Host header with every SCSP request to support web servers or server farms that host multiple domains. Your client might use an HTTP proxy that modifies the Host header, but the Swarm domain name matches the original Host header. In that case, an HTTP proxy copies the original Host header into another header, typically X-Forwarded-Host. Examples: ['X-Forwarded-Host', 'Host', 'X-ProxyForward-Host'] ['Host']
scsp.falseStartTimeout	240	In seconds, 0 to disable; defaults to 4 minutes. How long to wait to receive the first byte before timing out and disconnecting.
scsp.filterResponseBlacklist SNMP: filterResponseBlacklist	[]	Which headers to remove from HTTP responses. List is comma-separated and case-insensitive. For example: ['Castor-System-Path', 'Castor-System-Owner']
scsp.filterResponseHeaders SNMP: filterResponseHeaders	none	Swarm will filter response headers according to the given method. Allowed values: 'none', 'blacklist', 'whitelist'.
scsp.filterResponseWhitelist SNMP: filterResponseWhitelist	[]	Which headers to retain in HTTP responses, removing all others. List is comma-separated and case-insensitive. For example: ['Etag', 'Last-Modified']
scsp.idleDisconnectTimeout	14400	In seconds, 0 to disable; defaults to 4 hours. How long to wait after receiving the last byte before timing out and disconnecting.
scsp.keepAliveInterval SNMP: keepAliveInterval	15	How many seconds to wait before sending successive chunked keep-alive bytes after a 202 Accepted response.
scsp.maxContextReplicas SNMP: maxcontextreplicas	16	Count. Sets the maximum number of replicas in this cluster for a context (domain or bucket) object.

scsp.maxReadTime	10800	SCSP read time limit in seconds; defaults to 3 hours. SCSP GET requests running longer than this value will be prematurely closed.
scsp.maxWriteTime	10800	SCSP write time limit in seconds; defaults to 3 hours. SCSP write requests running longer than this value will be prematurely closed.
scsp.port SNMP: scspport	80	Port number; defaults to 80. The port used by client applications to access cluster nodes with HTTP requests. This setting mus be the same on all nodes in the same cluster.
scsp.replicateOnWrite SNMP: autoRepOnWrite	TRUE	Enabled by default. Improves content integrity by requiring a replica to be written in order for the POST, PUT, COPY, or APPEND request to succeed. Set to False to have the health processor manage creation of replicas after the write.
scsp.requireExplicitContextCreate SNMP: requireExplicitContextCreate	FALSE	When true, Swarm requires creation of a context (domain or bucket) to include the 'Content-type: application/castorcontext' header. Enable the option to protect against content being erroneously written as context objects, which hurts performance.
scsp.validateOnRead	FALSE	Disabled by default. Enable to force Swarm to validate the object's contents before returning successful read responses to client requests. Although validation can be specified on a per-read basis, this setting forces all reads to use validation. During the read from the disk, the content hash is computed. If the hash is wrong, indicating logical disk corruption, the socket will be closed before the last block is transmitted, forcing an error to the client. Note that using this option creates additional CPU load on the node.
search.caseInsensitive	FALSE	Whether metadata fields should support case-insensitive searching. If true, then all custom metadata will be indexed to support only case-insensitive searching.

search.pathDelimiter	/	Which character to use for parsing directory paths from object names, such as '2018/Q4/snapshot.pdf'. Defaults to forward slash: /
security.administrators	{'admin': 'ourpwdofchoicehere'}	One or more username:password pairs. Sets credentials for who can administer the cluster via the Swarm UI. If the value includes the snmp username, remove it from here and update snmp.rwCommunity with its password. Example: {'admin': 'adminpassword','admin2':'adminpassword2'
security.noauth	TRUE	[deprecated] To enable native Swarm authorization, set to False.
security.operators	{}	One or more username:password pairs. Sets credentials for who can view the Swarm UI. I the value includes an snmp username, it is ignored; remove it from here and update snmp.roCommunity with its password. Example: {'operator': 'operatorpassword','operator2': 'operatorpassword2'}
security.secureLogging SNMP: secureLogging	FALSE	Enable to prevent logging of the details of a client request. This option results in short, secure log messages.

snmp.getnextskips	['35', '36.20', '36.21', '36.22', '36.23', '36.25', '37.11.8', '38', '41', '55', '57', '58', '61', '63', '64', '65', '66', '68', '69']	List of OIDs to be skipped on output. To protect cluster performance, this setting causes the snmpwalk of the entire CASTOR MIB to skip several large, detailed tables in SNMP groups. The default list of OIDs causes a top-level snmpwalk to skip the groups or tables under clusterConfig, responseHistogramTable, hp, clusterdata, indexer, configVariableTable, castorFeeds, feedVolTable, performance, and recoveryTable. You can add or remove OIDs to control which sections of the MIB are returned by an snmpwalk. Enter values as strings in numeric form, relative to the Castor OID, .1.3.6.1.4.1.24659.1. Example: ['35', '37.11.8', '38', '41', '55', '57', '58', '61', '63', '64', '65', '66', '68', '69']
snmp.roCommunity	public	String. Password for the SNMP read-only community. If security.operators includes the snmp username, remove it and update the password here.
snmp.rwCommunity	ourpwdofchoicehere	String. Password for the SNMP read-write community. If security.administrators includes the snmp username, remove it and update the password here.
snmp.timeout SNMP: snmpTimeout	5	In seconds, 1-60. The snmpget, snmpset, and snmpwalk timeout for Swarm and Watchdog.

Node (Chassis) Settings

Note

Most of these settings (those without SNMP names) require a reboot of the affected machine (chassis) to take effect; the Swarm UI flags such settings with a restart icon.

Each physical or virtual machine is one Swarm node. On reboot, the local config file is read and the node inherits the settings from that file.

Name SNMP Name	Default	Description Examples
cache.expirationTime	600	In seconds; defaults to 10 minutes. Set 0 to disable. How long to hold an object after its last access.

cache.maxCacheableSize	1048576	In bytes, defaulting to 1 MB. The largest object that can be stored in the content cache. If increased to greater than 5 MB, then scsp.readBufferAllowance must be increased to the same value.
cache.percentage	10	Percentage, 0-100; set 0 to disable. How much I/O buffer memory to reserve for the content cache, which improves access to active content by storing it in geographically proximate locations. The reserve is reported when the node starts up: 'MAIN ANNOUNCE: Memory allocation at startup.' For best performance, especially with writing named objects, do not disable the content cache unless directed by Support.
cache.realmStaleTimeout	600	In seconds, 60 or higher. How long before the security user list cache for domains is cleared. Lower this value if user lists update frequently.
cip.queryTimeout	0.03	In seconds. How long after booting that the cluster will initially wait for node replication bids. Once the cluster is running, bid wait times are calculated dynamically based on response times. For clusters with network latency, the initial wait time may need to be increased until the cluster can correctly calibrate.
cip.readBufferSize	1048576	In bytes. The size of the multicast UDP socket read buffer.
console.expiryErrInterval	10	Number of days before the cluster license expires to generate an error as a log message and a console indicator.
console.expiryWarnInterval	30	Number of days before the cluster license expires to generate a warning as a log message and a console indicator.
console.indexErrorLevel	90	Percentage, 0-100. How much index utilization will generate an error as a log message and a console indicator.
console.indexWarningLevel	80	Percentage, 0-100. How much index utilization will generate a warning as a log message and a console indicator.
console.port	90	Which port Swarm uses to listen for requests. All nodes in the same cluster must be set to the same port. When deploying Swarm into untrusted network environments, firewall this port so that only administrators can access it.
console.reportStyleUrl		The URL for the path to the stylesheet and image files for configuring Swarm console. Example: http://10.10.15.32/css/swarm-reports.css
console.spaceErrorLevel	10	Percentage, 0-100. How much cluster capacity remaining will generate an error as a log message and a console indicator.

console.spaceWarnLevel	25	Percentage, 0-100. How much cluster capacity remaining will generate a warning as a log message and a console indicator.
console.styleUrl		The URL for the path to the stylesheet and image files for configuring the Swarm console. Example: http://10.10.15.32/css/swarm.css
disk.defragUntilPercentage	0.8	Ratio, 0.0-1.0. The portion of known unused space that, when untrapped, will stop the disk defrag process.
disk.enableMultipath	FALSE	Whether to enable support for Device Mapper Multipathing (DM-Multipath). Enabling multipath provides I/O failover and load balancing for storage devices.
disk.encryptNewVolumes	FALSE	Whether to encrypt new Swarm volumes. Enabling encryptNewVolumes means that any newly-formatted Swarm volume will be encrypted
disk.encryptionCipher	aes-xts-plain64	The encryption cipher to be used when setting encryption for new Swarm volumes. Supported values are aes-xts-plain64 and aes-cbc-essiv
disk.encryptionHash	sha512	The encryption hash algorithm to be used when setting encryption for new Swarm volumes. Supported values are sha256 and sha512.
disk.encryptionIterationTime	5000	In seconds. The maximum amount of time to be spent while iterating to generate an internal LUKS key from a Swarm encryption key, which will be used when setting encryption for new Swarm volumes.
disk.encryptionKeyPrimary		The mnemonic name of the encryption key to use for encrypting new Swarm volumes. Do not use quotes. For this key to be used, disk.encryptNewVolumes must be set to True. Example: cluster_key_5_15_2016
disk.encryptionKeySize	512	The size of the internal LUKS key to be used when setting encryption for new Swarm volumes. Supported values are 128, 256, and 512.
disk.encryptionKeys	{}	A comma-separated list of mnemonic name and encryption key pairs, used for accessing encrypted Swarm volumes. Example: {'cluster_key_5_12_2015': 'de3498245ce8bf89', 'cluster_key_5_15_2016': 'a24f8ec391ab3341'}

disk.ioErrorToRetire	2	Count. How many consecutive I/O errors (no more than disk.ioErrorWindow seconds between each error) that will force a volume to retire.
disk.ioErrorTolerance	200	Count. How many I/O errors are tolerated, past which the volume is taken offline immediately. Swarm then marks the volume as Unavailable and initiates both the volume recovery process (FVR) and the erasure coding recovery process (ECR) to relocate all the volume's objects.
disk.ioErrorWindow	172800	In seconds; defaults to 2 days. The length of time after which an I/O error is forgotten, if no other errors followed and the volume's state is OK. Works with disk.ioErrorToRetire to control when volumes are retired. The default values means that if more than one error occurs within 2 days, the volume is retired.
disk.minGB	64	How many GB a device must have to be eligible for automatic storage volume assignment with volumes = all. Set to 0 to include all disk devices.
disk.smudgesToRetire	4	How many soft errors (smudges) over the life of a volume will trigger Swarm to retire the volume. A soft error occurs when the health processor does not get the expected data when validating the object but the disk gave no explicit I/O (hard) error. Set to 0 to disable the automatic retire.
disk.standbyTimeout	360	In seconds. How long until an idle disk spins down automatically.
disk.volumes SNMP: vols		Required. Specifies the volume storage devices for Swarm to use. Valid entries: all, or a space-separated list of Linux volume identifiers, such as /dev/sda, /dev/sdb. all (recommended) is required for hot plugging and lets Swarm to use all volumes larger than disk.minGB. If a node is shut down longer than disk.obsoleteTimeout, all of its volumes are stale and cannot be used unless you force a volume remount by adding the :k (keep) policy option modifier. To specify the size, add a modifier with units: vols1:100m vols2:250g. Examples: all /dev/sda /dev/sdb
feeds.maxMem	100000	In bytes. The maximum memory allowed per feed, for queue management.
feeds.retry	[30, 300, 1200]	In seconds. The progressive number of retry attempts by the plug-in, when blocked. Example: [60, 60, 60, 3600]

feeds.statsReportInterval	300	In seconds. How frequently to report statistics.
health.parallelWriteTimeout	0	In seconds; defaults to 0 (disabled). How long before an uncompleted multipart upload times out and is aborted, deleting all uploaded parts. For SwarmNFS, must be enabled with a non-zero value, such as 1209600 (2 weeks).
health.startDelay SNMP: hpStartDelay	900	In seconds; defaults to 900 seconds (15 minutes). How long after a node starts up to begin Health Processor checking and recovery processes. This option creates a grace period for the remaining nodes to stabilize in the cluster, which is useful in situations in which an entire cluster must be shut down and restarted.
license.url	<Swarm default 2T license>	The location and name of the Swarm license file, caringo/license.txt. Can be a pathname or a URL. To use the default 2 TB license, you must keep the default location. Example: http://10.10.15.32/config/swarm-license.txt
log.obscureUUIDs	FALSE	Whether to obscure UUIDs from displaying in logs. Set the value to True to abbreviate the UUID, if indicated by your security requirements.
mdns.readBufferSize	1048576	In bytes. The size of the read buffer for the multicast UDP socket.
network.dnsDomain		Optional. The domain name that will be used. Ignored unless network.ipAddress is set. To enable name resolution when using static IP addresses for Swarm nodes, configure both network.dnsServers and network.dnsDomain. This allows the resolv.conf file to be created for name resolution. Example: yourcompany.com
network.dnsServers		Optional. The DNS servers that will be used. Ignored unless network.ipAddress is set. To enable name resolution when using static IP addresses for Swarm nodes, configure both network.dnsServers and network.dnsDomain. This allows the resolv.conf file to be created for name resolution. Examples: 8.8.8.8 8.8.4.4 84.200.69.80 84.200.70.40
network.gateway SNMP: gateway		Optional. The default gateway IP address in the subnet. Ignored unless network.ipAddress is set. Example: 10.10.12.253
network.icmpAcceptRedirects	TRUE	Determines if the node accepts routing information from ICMP redirect responses.

network.igmpVersion	2	Range, 1-3. The IGMP (Internet Group Management Protocol) version that the Linux kernel will use for host membership queries.
network.ipAddress SNMP: ipaddress		The static IP address for a node to use, or blank to use DHCP. Multiple IP addresses are no longer supported. Example: 10.10.12.1
network.iptablesFileUrl		Optional. Location (URL) of Linux firewall rules to apply. When specified, Swarm transmits the rules without validation to the 'iptables-restore' command before starting the storage node processes. Example: http://10.10.15.32/config/swarm-iptables
network.mtu	1500	In bytes. Sets the maximum transmission unit (MTU) that Swarm accepts. Set to a higher value to use jumbo frames. Before you change the default value, verify that the node's network interfaces and all other network hardware support the selected MTU; otherwise, the nodes might not be able to replicate objects or communicate.
network.netmask SNMP: netmask		Optional. Sets the IP network mask for a node. Ignored unless network.ipAddress is set. Examples: 255.255.255.0 255.255.0.0
network.timeSource SNMP: timeSource		Required. The IP address of an NTP server. If the nodes cannot access public NTP servers, they will time out waiting for a connection and automatically restart. Only use trusted NTP servers, whether they are dedicated hardware solutions in your internal network or external, public servers that the nodes can access over the network. Examples: 0.north-america.pool.ntp.org 10.12.14.14
node.archiveMode SNMP: archiveMode	FALSE	Disabled by default, which is the normal operating state. Set to TRUE to change the node to archive mode, where it remains idle in low-power mode without participating in cluster activity until its capacity is needed. This setting is useful for proactively provisioning new nodes into the cluster before they are needed.

node.subcluster SNMP: subcluster	default	Specifies the name of the subcluster to which the chassis belongs. Names can have no more than 16 characters and no special characters, such as quotes and hyphens. Example: subcluster1
shutdown.gracePeriod	120	In seconds; defaults to 2 minutes. How long to allow ongoing SCSP requests to complete during shutdown.
snmp.sysContact	Unspecified	The value for the SNMP system contact, SNMPv2-MIB::sysContact. Must be a valid email address in 7-bit USASCII in one of these forms: Name <email@domain> First Last <email@domain> Example: admin@yourcompany.com
snmp.sysLocation	Unspecified	The value for the SNMP system location, SNMPv2-MIB::sysLocation. Example: rack3
snmp.sysName	Unspecified	The value for the SNMP system name, SNMPv2-MIB::sysName. Example: Joe Administrator
startup.certificates		Public certificates to add to cert bundle.

Persisted Settings

A subset of Swarm configuration settings are persisted settings, which are stored in a Settings object in your cluster if you have any domains or have ever changed settings in the Storage Management UI (or legacy Admin Console). These special settings persist across reboots, regardless of how you may have updated your configuration (node.cfg/cluster.cfg) files.

> Important
>
> The Settings object persists and overrides the configuration files, storing both settings and passwords for the cluster. See Swarm Passwords.

Best practice – Always change settings via the Storage Management UI, rather than through the configuration files. There are several benefits to this practice:

- No reboot required. You do not need a full reboot for the configuration change to take effect.
- Updates persisted settings. The changes are stored directly in the Settings object.
- Only update in one place. You only need to update persisted settings on one node: Swarm will propagate the changes to all the other Settings objects in your cluster.

> SNMP version
>
> Swarm supports SNMP version 2 only.

Here is an example SNMP set command that changes the string that is the policy for cluster-wide versioning:

```
snmpset -m +CARINGO-CASTOR-MIB -v2c -M +/usr/share/snmp/mib2c-data
-cPASSWORD -OQs
 {cluster} policyVersioning s "allowed"
```

Here is an example that changes an integer value:

```
snmpset -m +CARINGO-CASTOR-MIB -v2c -M +/usr/share/snmp/mib2c-data
-cPASSWORD -OQs
 {cluster} healthExamDelay i 30
```

Note
<cluster> in a URL stands for <host>[:<port>], where host is a fully qualified domain name or IP address, plus a port number if other than `80`. If the `Host` header does not match the domain name, override it with the `domain=` argument.

Listed below are the special Swarm settings that are persisted, with the SNMP Name to use. Those that have an entry for the Admin Console are settable there via the field indicated, but all are settable via the Storage Management UI.

Setting Name (settable in legacy Console)	SNMP Name	SNMP Default Value
bidding.relocationThreshold	relocationThreshold	
cip.queryRetryMultiplier	queryRetryMultiplier	
console.domains	Not settable.	
console.messageExpirationSeconds (9.1)	messageExpirationSeconds	
ec.conversionPercentage	ecConversionPercentage	
ec.segmentConsolidationFrequency (9.1)	ecSegmentConsolidationFrequency	
health.defragInterval	healthDefragInterval	
health.offloadPauseInterval	healthOffloadPauseInterval	

health.relocationVolumeFillRate	hpRelocationVolumeFillRate	
health.replicationMulticastFrequency	repMulticastFrequency	
health.replicationUnicastFrequency	repUnicastFrequency	
index.optimize404	overlayOptimize404	
index.overlayEnabled	overlayIndexEnabled	
index.ovMinNodes	overlayMinNodes	
log.host	logHost	
log.level	logLevel	
log.port	logPort	
metrics.period	metricsPeriod	
metrics.port	metricsTargetPort	
metrics.targets (9.1)	metricsTargetHost	
network.igmpTimeout	networkIGMPTimeout	
policy.ecEncoding	policyECEncoding	`policyECEncoding s "unspecified anchored"`
policy.ecMinStreamSize	policyECMinStreamSize	`policyECMinStreamSize s "1MB anchored"`
policy.replicas	policyReplicas	`policyReplicas s "min:2 max:16 default:2 anchored"`
policy.versioning	policyVersioning	`policyVersioning s "allowed"`
power.savingMode	powerSavingMode	
power.sleepAfter	sleepAfter	
power.wakeAfter	wakeAfter	
recovery.completedRecoveryExpiration	completedRecoveryExpiration	
recovery.suspend	volumeRecoverySuspend	
recovery.suspendedVolumes	Not settable.	

recovery.volMaintenanceInterval	volMaintenanceInterval	
scsp.allowPutCreate	allowPutCreate	
scsp.autoContentMD5Computation (9.1)	autoContentMD5Computation	
scsp.autoRecursiveDelete (deprecated 9.3)	autoRecursiveDelete	
scsp.replicateOnWrite	autoRepOnWrite	
scsp.requireExplicitContextCreate (9.1)	requireExplicitContextCreate	
security.secureLogging	secureLogging	
snmp.timeout	snmpTimeout	

Replaced Settings

The following table lists obsolete settings and their corresponding new, qualified names. Be sure to update your configuration to use the new settings:

1. Update your `node.cfg/cluster.cfg` files.
2. Update your persisted settings via SNMP. See Persisted Settings.

> Warning
> Incorrectly renaming a setting can prevent the node or cluster from booting.

The following table lists settings alphabetically by the deprecated names.

Old Setting	New Setting
autoRepOnWrite	scsp.replicateOnWrite
autoValidateRead	scsp.validateOnRead
chassis.processes	No longer needed, v10.0
cipTTL	cip.ttl
cluster	cluster.name
cluster.proxyPort cluster.proxyIPAddress	cluster.proxyIPList
cluster.settingsUuid	No longer needed
consolePort	console.port

consoleReportStyleURL	console.reportStyleUrl
consoleStyleURL	console.styleUrl
defreps	policy.replicas
disk.automaticFormat	No longer supported
ec.encoding	policy.ecEncoding
ec.minStreamSize	policy.ecMinStreamSize
ec.subclusterLossTolerance	No longer needed, v10.0
hpStartDelay	health.startDelay
ipaddress	network.ipAddress
licenseFileURL	license.url
loghost	log.host
loglevel	log.level
logport	log.port
maxreps minreps	policy.replicas
networkMTU	network.mtu
realmCacheStaleTimeout	cache.realmStaleTimeout
repMulticastFrequency	health.replicationMulticastFrequency
repPriority	health.replicationPriority
repThreshold	bidding.relocationThreshold
scsp.minReplicas scsp.maxReplicas scsp.defaultReplicas	policy.replicas required parameters: min, max, default
scspport	scsp.port
security.noauth	No longer supported
snmpSysContact	snmp.sysContact
snmpSysLocation	snmp.sysLocation

snmpSysName	snmp.sysName
spaceErrLevel	console.spaceErrorLevel
volMinimumGB	disk.minGB
volPluginURL	disk.volumeIdentifyUrl
vols	disk.volumes
volStandbyTimeout	disk.standbyTimeout
volumeRecoverySuspend	recovery.suspend

Configuring Content Policies

- Working with Policies
- Replication Policy
- Erasure Coding Policy
- Versioning Policy

Working with Policies

To make fullest use of the rich content protection features of Swarm, you will design and apply precise policies for where and how to apply protections in your implementation. Policies define how the data being uploaded to your Swarm cluster should be stored, managed, and protected.

How it works

Cluster-level policies reside in configuration settings and let you define cluster-wide and cluster-specific policies for how Swarm implements its content protection features. You can layer these baseline cluster policies with context-level (domain and bucket) policies, which reside in designated headers in the context objects themselves. Policies are not lifepoints: they do not have built-in lifetimes, nor do they specify transitions from one policy to another over time. Policies are only changed by explicit updates.

Types of content policies

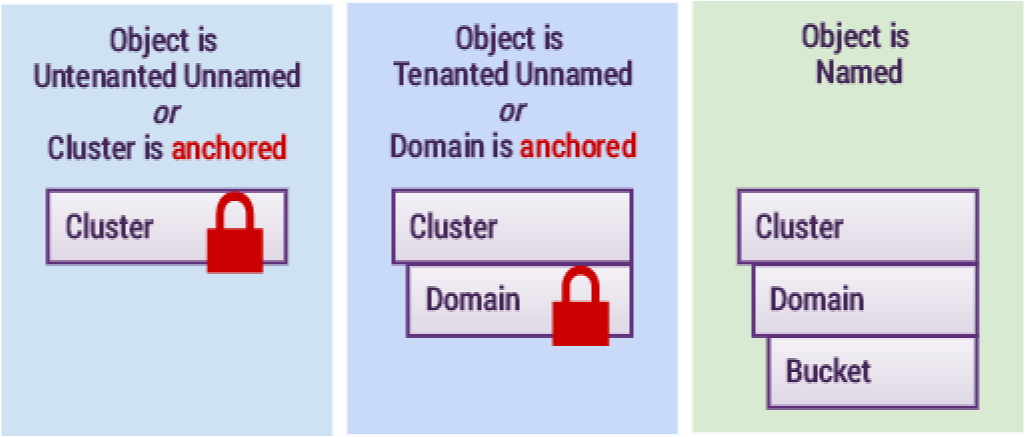
You can set policies for all of these content protection options, specific to the context you need:

- Replication - how many default, minimum, and maximum number of replicas to keep for the objects in this cluster/domain/bucket
- Erasure Coding EC - whether to enable or specify the EC encoding (`k:p`) to use for the objects in this cluster/domain/bucket
- Object Versioning - whether to enable, disable, or suspend versioning for the objects in this cluster/domain/bucket

How policies resolve

To resolve overlapping policies, Swarm begins policy evaluation by eliminating contexts (domain and bucket) that do not apply, for one of two reasons:

- because of the "`anchored`" parameter, which overrides lower-level policies
- because of the type of object, which restricts it to certain contexts:



Swarm (1) selects the anchored policy if it exists, or else (2) checks the relevant contexts from the bottom up and selects the first policy it finds.

> Note
> Within a given request to a context (domain or bucket) object, if more than one policy header of the same type is written, the last one written is honored.

Setting policies by cluster

You can set cluster-specific policies for domains and buckets by including the cluster name parenthetically and separating them with a comma. Swarm looks for and takes any cluster-specific policy values, using the default value only if it finds no match.

For example, you many want a policy (such as versioning) to be enabled in the main cluster but disabled in the target of your replication feed. This is how you enable a feature in your main cluster (myCluster) but disable it in your one-way replication cluster (myRemote):

```
Policy-{feature}: enabled (myCluster), disabled (myRemote), disabled
    or
  Policy-{feature}: enabled (myCluster), disabled
```

> Required
> Always end a multi-part policy with the default value that you prefer.

Updating policies

Swarm cluster policies are all persisted settings, so use the snmpset command on the cluster's settings object to change and persist the policies cluster-wide. Domain and bucket policies are saved as headers on those objects.

> See Using SNMP with Swarm.

Replication Policy

Deprecated
Swarm 8.2 added policy support for replication, which extends your ability to set replication for contexts (domains and buckets) as well as clusters. This changed three settings:

- scsp.minReplicas, scsp.maxReplicas, scsp.defaultReplicas are deprecated, replaced by one combined setting: `policy.replicas: min:# max:# default:# [anchored]`

Use of the old settings will generate errors. When you upgrade, remove these settings from your configuration files and use the new setting going forward.

Setting	Default	
policy.replicas	min:2 max:16 default:2 anchored	Required. Must include all three parameters, min, max, and default. If a value is missing or out of range, Swarm returns 400 (Bad request). Add `anchored` to prevent overrides of the policy within the cluster. Remove it to enable domain and bucket policies. Persisted setting. The SNMP MIB name is policyReplicas.

> See Implementing Replication Policy for how to create custom replication policies on specific domains and buckets.

Erasure Coding Policy

Deprecated
Swarm 8.1 added policy support for EC encoding, which extends your ability to set encoding for contexts (domains and buckets) as well as clusters. This changed two settings:

- ec.encoding is deprecated, replaced by policy.ecEncoding
- ec.minStreamSize is deprecated, replaced by policy.ecMinStreamSize

Use of the old settings in 8.1 will generate errors that are visible on the Swarm Admin Console. When you upgrade, remove these settings from your configuration files and use the new settings going forward.

> See Implementing EC Encoding Policy for how to create custom EC encoding policies on specific domains and buckets.

Setting	Default	

ec.conversionPercentage	0	Percentage, 1-100; defaults to 0, which disables conversion of old objects. A setting of 5 is recommended. How many objects that have become eligible to convert erasure coding will be converted each HP cycle. Persisted setting. The SNMP MIB name is ecConversionPercentage. Note This setting lets you lower the speed of conversion if you need to reduce load on the cluster. You can also increase the speed of conversion to hasten the processing of a large collection of eligible objects, but be sure to plan for that impact on the cluster.
ec.maxManifests	6	Range, 3-36. The maximum number of manifests written for an EC object. Usually, p+1 are written for a k:p encoding. Do not set above 6 unless directed by Support.
ec.minParity	1	Range, -1 for max(policy.replicas min - 1, 1). The minimum number of parity segments the cluster requires. This is the lower limit on p for EC content protection, regardless of the parity value expressed on individual objects through query arguments or lifepoints.
ec.protectionLevel	node	Either 'subcluster' or 'node'; case-sensitive. The protection level that determines the segment distribution needed for a successful EC write. Node-level protection ensures that each node in the cluster has no more than one EC segment for a given object. Subcluster-level protection ensures that erasure-coded data will survive the loss of one or more subclusters, to the level set by ec.subclusterLossTolerance.

ec.segmentSize	200,000,000	In bytes; defaults to 200 MB, with a minimum of 100 MB. The maximum segment size for each erasure-coded data segment in an erasure set before another erasure set is created. Given 5:2 encoding, this default allows a 1-GB object to be written using only one EC set, a 2-GB object using two EC sets, and a 3-GB object using three. Best practice: Increase your segment size for larger objects, to minimize the number of sets needed. Multiple EC sets take up physical memory because each of their segments require an index slot in memory. Therefore, if you know that you will be storing an object larger than 1 GB, increase ec.segmentSize so that fewer EC sets are needed to store it. That is, if your typical object will always be 2 GB rather than 1 GB, double the ec.segmentSize. Tip In addition to changing the cluster-wide default, you can override the segment size as needed by adding the segmentsize query argument to the write request.
ec.subclusterLossTolerance	-1	(Subcluster-specific) -1 or 1, for more or less protection, respectively. Specifies the subcluster protection level that ensures the EC segments are distributed over enough subclusters to prevent data loss. More (-1) distributes one segment per subcluster, which means that p subclusters could be lost without data loss; it is most useful for multi-server chassis in which each chassis is a subcluster by default. Less (1) distributes up to p segments per subcluster, spread across as many subclusters as possible, which means that one subcluster could be lost without data loss.
policy.ecEncoding	unspecified anchored	(Replaced ec.encoding in 8.1) The cluster-wide setting for the EC (erasure coding) encoding policy. Valid values: unspecified, disabled, k:p (a tuple such as 5:2 that specifies the data and parity encoding to use). Add `anchored` to prevent overrides of the policy within the cluster. Remove it to enable domain and bucket policies. Persisted setting. The SNMP MIB name is policyecEncoding.

policy.ecMinStreamSize	1MB anchored	(Replaced ec.minStreamSize in 8.1) In units of megabytes (MB) or gigabytes (GB); must be 1MB (default) or greater. The size that will trigger an object to be erasure-coded, if specified (by ecEncoding, lifepoint, query arg) and allowed by policy. Exception: If the request is chunked or a parallel write (which must be erasure-coded), then the ecMinStreamSize limit does not apply. Add `anchored` to prevent overrides of the policy within the cluster. Remove it to enable domain and bucket policies. Important For efficiency, policy.ecMinStreamSize is given in units of nMB and nGB, not bytes, as were the old settings. Be sure to convert (not copy) your old ec.minStreamSize value to the new setting that specifies units. Note that 1 MB = 1,000,000 bytes. Persisted setting. The SNMP MIB name is policyecMinStreamSize.

EC Protection Level

The EC protection level determines how strictly Swarm distributes erasure-coded segments across hardware groupings. Valid values are node and subcluster. The default value is node protection, which suffices for most use cases.

Regardless of the protection level you set, Swarm always makes a best effort to distribute segments as broadly as possible across your hardware, to protect your data. The protection level lets you control how Swarm enforces the segment distribution policy on write, which it does by failing the write and returning an error response to the writing application if the policy is not met (412 Precondition Failed).

> 'm' segments
>
> Erasure coding breaks the object into data segments (k) and computes parity segments (p) based on the content of the data segments. This results in m total segments (k + p = m), distributed to m different nodes (or subclusters) in the cluster.

Setting	Value	Description

ec.protectionLevel	node (default)	Makes Swarm write and validate that each node in the cluster has no more than one EC segment for a given object. If there are not enough nodes to accept the total number of segments (k + p = m), the write fails. If a node with a segment fails and there are no longer m nodes available, the cluster logs a warning but degrades gracefully to volume distribution, ensuring that each volume has only one segment. Important At node-level protection, there must be at least m nodes for the write to succeed. Subcluster distribution When using node-level protection, Swarm tries to distribute segments across configured subclusters, but it does not fail a write if it cannot. When the health processor checks the validity of each erasure-coded object, it will try to redistribute across subclusters, if needed.
	subcluster	Subcluster-level protection makes Swarm meet the required segment di stribution for a successful EC write. Use this protection level to protect your erasure-coded data from the loss of one or more subclusters. When this is specified, the ec.subclusterLossTolerance setting controls the degree of subcluster protection.
ec.subclusterLossTolerance	-1 (default)	More protection: Setting subclusterLossTolerance to -1 (default) lets you survive the loss of p (parity) subclusters without any data loss. It makes Swarm write and validate that each subcluster in the cluster has no more than one E C segment for a given object. If there are not enough subclusters to accept m segments, the write fails. That is, 5:2 erasure coding needs 7 subclusters (7 chassis) to write the object; if one of those chassis fails for any reason, writes will fail. If a subcluster storing a segment fails and there are no longer m subclusters available, the cluster logs a warning but degrades gracefully to node distribution, ensuring that each node has only one segment. Tip This configuration is most useful for multi-server chassis in which each chassis is a subcluster by default.

	1	Less protection: Setting subclusterLossTolerance to 1 lets you survive the loss of one subcluster without any data loss. It lets Swarm distribute up to p segments per subcluster such that loss of one subcluster will still keep the object is still accessible, without requiring as many subclusters (m or more). That is, 5:2 erasure coding needs 4 (not 7) subclusters, each with two segments. With this setting, loss of a subcluster can let you read but not write EC objects. For instance, with 9:6 erasure coding and 3 configured subclusters, each subcluster would get 6 or fewer segments. Losing any one of the three subclusters, Swarm could read an EC object via the segments remaining in the other 2 subclusters, but it could not write an EC object because of failing the precondition. That is, the cluster would have written 8 segments to one subcluster and 7 to the other, resulting in inadequate protection: loss of one of the remaining subclusters would make the object inaccessible.

After Swarm writes an object to the cluster, the health processor tries to maintain the requested protection level. If cluster resources become unavailable, it will degrade gracefully. When this occurs, the health processor logs errors, alerting you that the requested subcluster protection cannot be maintained and that your data may be at risk of subcluster loss.

Versioning Policy

Swarm 8.0 added policy support for versioning, which makes it possible to enable versioning for specific contexts (domains and buckets) after you configure the cluster to permit versioning of objects.

Setting	Default	
policy.versioning	disallowed	Defaults to disallowed, which prevents all versioning. • Set to allowed to enable context policies to enable versioning as needed. • Set to suspended to stop creation of new versions throughout the cluster but retain any existing historical versions of objects. • Set to disallowed to stop creation of new versions throughout the cluster and to remove all historical versions of objects. Persisted setting. The SNMP MIB name is policyVersioning.

> See Implementing Versioning for how to create versioning policies on specific domains and buckets.

- Recommendations for Erasure Coding
- Erasure Coding across Subclusters

Recommendations for Erasure Coding

- Number of nodes
- RAM per node
- Encoding level
- Performance considerations

While the cluster will fail a write request in a cluster that does not have at least k+p Swarm nodes available, the minimum number of required nodes is actually higher, to allow space for recovery in the event of a failed volume.

Number of nodes

- The minimum number of nodes to ensure base functionality is k+(p*2) .
- The recommended number for performance and load distribution is (k+p)*2 .
- 5:2 encoding requires at least 9 Swarm nodes, preferably 14 nodes.

RAM per node

Nodes using erasure coding will benefit from additional CPU cores (and Swarm multi-server processes to utilize them).

- The recommended minimum of RAM per node is 4 GB.

Encoding level

- Do not run Swarm using an EC level with fewer than 2 parity segments, except in cases of transient data or data that can be reproduced or regenerated easily.
- Good baseline encoding is 4:2 or 5:2; this ensures data protection equal to or better than replication, with a smaller footprint.
- Subclusters have specific encoding considerations.

Performance considerations

Write throughput for large objects that are erasure-coded is approximately equivalent to replicated objects written with Replicate on Write. Smaller objects are less performant due to the overhead of creating the parity segments relative to the time to write only the content body.

- Parity Count. The biggest impact on performance for similar-sized objects is the parity count: the smaller the number of required parity segments, the faster the write performance.
- Segment Size. If you can predict the size of the objects written to the cluster, you can increase the default segment size to reduce the number of EC sets it takes to encode the object, thereby increasing performance. For example, for a 2 GB object with 5:2 encoding, increasing the segment size to 400 MB will ensure that the object fits in one EC set.
- Write Threads. Erasure-coded objects require that more Swarm nodes participate in each write request. As a result, the number of client threads each node can service at a time is less than with replicated objects. For planning purposes, assume that each Swarm node can service 4 erasure-coding client write threads at a time before the cluster begins to return 503 (Service Unavailable) responses, indicating the cluster is too busy to service more requests. That threshold may differ depending on the size of the cluster and the encoding you use.

Erasure Coding across Subclusters

In addition to spreading segments out across nodes (so that no two segments for the same erasure set are on the same node), Swarm automatically distributes segments evenly across subclusters. If it is configured correctly, this distribution keeps your data is protected even if an entire site fails.

In a subcluster configuration, your data is safe when the following is true:

```
Subclusters remaining / Total # of subclusters >= k data-segments / (k
data-segments + p parity-segments)
```

Example: Subcluster setting that does not ensure full recovery

With two subclusters containing four nodes each and EC set to 5:2 for seven total segments, one subcluster would have three segments and the other would have four segments. The number of segments in each subcluster is greater than the parity count (in this example), so the object would NOT be fully recoverable if you lost an entire subcluster. Using the formula above to double check shows that the data is NOT safe if a single subcluster is lost:

```
1/2 IS NOT >= 5 /(5+2)
```

Example: Subcluster setting that does ensure full recovery

With 3 subclusters containing 5+ nodes each and EC set to 9:6 for a total of 15 segments, each subcluster would have five segments. If you lost an entire subcluster, you would only lose five segments, which is less than the set parity level, and therefore you would be able to recover the object.

Using the formula above to double check shows that the data is safe if one of the three subclusters is lost:

```
2/3 >= 9 / (9+6)
```

The following table compares these examples.

EC Encoding	# Nodes required to write an EC object	Max # Nodes that can fail and still read the object	Protected with loss of 1 of 2 subclusters?	Protected with loss of 1 of 3 subclusters?
5:2	7	2	NO. With 7 segments, one subcluster would have 3 segments and the other would have 4 segments, both over the maximum allowed.	NO. With 7 segments, one subcluster would have 3 segments and the other two would have 2 segments each. If the subcluster with 3 objects went offline, the object would not be recoverable because that value is over the maximum allowed.
9:6	15	6	NO. With 15 segments, one subcluster would have 8 segments and the other would have 7 segments, both over the maximum allowed.	YES. With 15 segments, each subcluster would have 5 segments, which is less than the maximum allowed.

Configuring Content Integrity Settings

Use the following set of configuration settings to enforce your content integrity policy.

- replicateOnWrite
- replicationPriority
- validateOnRead

replicateOnWrite

This [scsp] option allows an administrator to force a second replica of an object to be written to a different node prior to returning a success status to the client. If either write operation fails, the client receives an error status.

> See Configuring Replicate On Write.

replicationPriority

This [health] option allows an administrator to control how aggressively to handle asynchronous replications of an object after it is initially written into the cluster.

The default setting is 1, which gives the replication task equal priority with user requests. Setting the parameter to 2 makes the replication secondary to handling user requests.

The scsp.replicateOnWrite option can be used to make replication synchronous, which is the highest priority.

> Note
>
> If you use both replicateOnWrite and replicationPriority=1 at the same time, the ROW semantics apply to the first two replicas and any additional replicas (if reps > 2) are written with priority 1.

validateOnRead

This [scsp] option allows an administrator to require validation of all content reads before returning a successful read completion.

Although this option can be specified on a per-read basis, setting the value to 1 in the configuration file forces all reads to use validation.

During the read from the drive, the content hash is computed. If the hash is wrong, indicating logical drive corruption, the socket will be closed before the last block is transmitted, forcing an error to the client.

> Note
>
> Using this option creates additional CPU load on the node.

Configuring ROW Replicate On Write

When you execute a Replicate On Write (ROW) command for data protection, Swarm creates all of the replicas requested at once, in parallel. Using ROW, objects are safe (replicated on other nodes) even if a disk failure occurs immediately after a write or update completes.

ROW versus HP for replication

Without ROW, the Health Processor (HP) manages all replication in the background. When you write an object, the HP checks its replication constraint. When you update an existing object, the HP creates a duplicate of the updated version and deletes the older replicas on the cluster nodes.

With ROW, Swarm creates another instance of the object on another node immediately, as part of the write. ROW ensures that two or more object replicas (instances) exist in the cluster before the client write request is completed. If the object includes a constraint for creating more copies, those additional copies are made during the normal health checking process. While ROW may temporarily restrict client responsiveness as objects are created and replicated at the same time, your objects are protected from a single volume failure right away.

ROW benefits	ROW effects
• Immediately protects an object from a single-volume failure rather than waiting for the Health Processor to duplicate the object during normal operations. • Quickly deletes older replicas to ensure that all versions are current. • Quickly invalidates cached domain versions in the cluster so that the latest version is implemented.	• Takes a bit longer to write than a single replica. • Is more likely to fail, because of the additional preconditions.

ROW commands

ROW is enabled by default. You can issue ROW-related commands in these ways:

Configuration	In node.cfg, you can disable ROW with this setting. (See Settings Reference.) `scsp.replicateOnWrite = false`
SNMP	To disable the setting without restarting the cluster, set `autoRepOnWrite=0`. (See Persisted Settings.) Important Once `autoRepOnWrite` has been changed via SNMP, it is stored in the persistent settings. Any future changes must be via SNMP or the Swarm UI to override that setting.

Query Arg	Use a `replicate` query argument whenever needed to override the cluster-wide ROW configuration. (See WRITE with Replicate ROW.)
	ROW enabled: To override the cluster defaults, use the query argument with an integer. The most common usage, `replicate=1`, allows the write to succeed with only one instance of the object, which effectively disables ROW for the write:
	`POST /?replicate=1`
	ROW disabled: If you have ROW disabled in the cluster, you can achieve ROW by adding the query argument with `immediate` or `full` to each write:
	`POST /?replicate=immediate`
	`POST /?replicate=full`
	Note
	When you are creating or updating a bucket, the `replicate=immediate` option quickly invalidates cached bucket versions in the cluster so the latest version will be implemented in the cluster. It also prevents subsequent permission errors because out-of-date permissions are used from the prior version.

Configuration settings

Following are the configuration settings that control replication for your storage cluster.

scsp.replicateOnWrite

ROW is enabled by default. You can disable ROW in your storage cluster by changing the setting to `scsp.replicateOnWrite = false`.

The settings scsp.replicateOnWrite and scsp.defaultROWAction are evaluated together. When scsp.replicateOnWrite is true (the default), Swarm withholds sending the write verification to the client until the number of replicas specified by the scsp.defaultROWAction setting (default 2) have been created successfully.

> Important
>
> Any query arguments in the request override the configuration settings.

scsp.defaultROWAction

scsp.defaultROWAction is a positive integer or full (a special keyword). Full indicates that the number of replicas is the number determined by the greater of the object's reps lifepoint or the policy.replicas default setting for that object. The limits defined by the cluster policy.replicas min and policy.replicas max settings are enforced on the number of replicas created in a ROW request as well.

The replicate query argument serves the same purpose with the similar acceptable values.

> See WRITE with Replicate ROW.

policy.replication

Your replication policies (`policy.replicas: min:# max:# default:# [anchored]`) and the policy evaluation process affect how Swarm applies ROW. These settings define the baseline range of allowable replicas:

Replication Configured	Effect	Value	Description
scsp. replicateOnWrite	lower limit	true | false	Swarm starts evaluation with the value of replicateOnWrite: 1 replica if false and 2 replicas if true (the default).
policy.replicas max	upper limit	min ≤ n ≤ 20	Swarm determines how many replicas should be made for an object and then limits the SCSP request to that number. The the replication policy `default` is the key value: if it is 2 and the request asks for more, it gets only 2 on the write.

When the replicate query argument is used, the remaining policy.replication parameters have these effects:

Replication Requested	Meaning	Effect on evaluation	Evaluates to	Description
replicate=immediate	For use if ROW is disabled in the cluster. Only two replicas must be created before Swarm sends the response; if more replicas are required, those additional replicas are created after the response.	lower limit	`2`	Only 2 replicas get created synchronously.
replicate=full	For use if ROW is disabled in the cluster. All of the required replicas must be created before Swarm sends the response.	default value	`policy.replicas default`	The policy.repl icas default ev aluated for the object determines the number of replicas that Swarm will create synchronously.

replicate={integer}	How many replicas must be created synchronously for this write, overriding cluster settings (within constraints). If policy.replicas max or the lifepoint reps is less than this integer, Swarm uses the smaller of those values. Tip Use `replicate=1` to override ROW and achieve the fastest possible writes, for situations that require that.	default limit	{integer} ≤ `policy.replicas default`	The policy.replicas default evaluated for the object serves as an upper limit on {integer} to determine the number of replicas that Swarm will create synchronously.

> Note
>
> Neither the replication policy nor the replicate query argument has any effect on erasure-coded objects. For a k : p encoded EC object, Swarm will keep p+1 manifests, up to a limit of `ec.maxManifests`.

> See Configuring Content Policies and Implementing Replication Policy.

Configuring Volumes Options

- disk.volumes setting
- device option
- policy option

The **`disk.volumes`** option in the node or cluster configuration file specifies the volumes used by Swarm. This specification includes the device names and optional flags for handling these volumes. Only one **`disk.volumes`** parameter is allowed in a configuration file.

> Warning
>
> Swarm erases any non-Swarm data on all the volumes it uses. For best results, run Swarm only on nodes that are free of non-Swarm data.

disk.volumes setting

Best practice — Use **`disk.volumes = all`**, which enables Swarm to use all volumes larger than the configured **`disk.minGB`** (64 GB by default). You can configure Swarm to format and mount smaller drives by decreasing the value of `disk.minGB`. If you are booting Swarm from a USB flash drive, the drive is automatically excluded from the volume list. Additionally, **`disk.volumes = all`**:

- Supports hot plugging
- Supports exceptions (`all except`)
- Supports `:k` (keep policy)

volumes syntax

```
disk.volumes = volume-specification
volume-specification ::== all-volumes | volume-list | ''
all-volumes ::== 'all' [ ':' policy ] [ space 'except' space device-list
]
device-list ::== device [ space device [ ... ] ]
device ::== Linux-device-path
space ::== space or Tab character and not the word space
policy ::== 'k'
```

Example disk.volumes entries

```
disk.volumes = all
disk.volumes = all:k
disk.volumes = all except /dev/hda
disk.volumes = /dev/sda /dev/sdb
disk.volumes = /dev/hda:k /dev/hdc:k
disk.volumes =
```

No volumes — If disk.volumes is set to an empty string, you are specifying a diskless machine, so Swarm will not mount any volumes. If the disk.volumes setting itself is absent, Swarm will mount all available volumes (equivalent to disk.volumes=all).

device option

The device component is either the keyword all or the Linux device path string for the drive. When using the keyword `all`, do not include any other device path specifications unless they follow `except`.

Best practice — Use `disk.volumes = all` and avoid Linux device paths. However, you can exclude specific volumes from being formatted and used by Swarm by listing them after `except`:

```
disk.volumes = all except /dev/sda dev/sdb
```

> Important
> Only use `except` with `disk.volumes = all`.

Example Linux device paths

- /dev/hda
- /dev/sda
- /dev/sdb

Older IDE drives, also known as Parallel ATA, EIDE, ATA-33, ATA-66, ATA-100, or ATA-133, use hd device names. These drives are configured as master or slave devices on each IDE controller. Typically, the master devices are /dev/hda and /dev/hdc, while the slave devices are /dev/hdb and /dev/hdd.

SCSI, SAS, and SATA drives typically use sd device names. The device letters are assigned sequentially in the order in which the drives are discovered starting at /dev/sda. The hardware report in the utility menu will show the actual names in use on a node.

If an invalid device name is specified in the device component, the node log indicates an error during the format operation. Incorrectly-specified volumes are not used.

policy option

The policy option lets you instruct Swarm how to handle a volume. Currently, these handling features involve the formatting characteristics of the physical device.

The format policy allows you to override the default volume expiration behavior by specifying the :k (keep) policy.

> See Returning a Stale Volume to Service.

Configuring Power Management

Swarm includes an adaptive power conservation feature that supplements Swarm's naturally green characteristics. This power-saving mode, also referred to as Darkive™, spins down disks and reduces CPU utilization after a configurable period of inactivity.

> Note
> The configuration options for power management apply to the entire cluster.

How Power Management Works

The power-saving mode causes a node with no incoming SCSP requests (from clients or other Swarm nodes) in the last configurable power.sleepAfter seconds to change to an Idle status and pause its health processor. There is usually a delay between the node Idle status and its volumes Idle status because in-process replications are performed between nodes even after they are idle, to ensure full data protection. After all queued activity is completed, the disks will eventually spin down (if supported by the disk manufacturer) and display as Idle if no further activity causes disk I/O.

When it appears in the Swarm UI or Console, Idle has different meanings for a node and volume:

- Idle node. A node with no SCSP activity for a specified length of time. In an idle state, a node's health processor pauses, so an idle node is more likely to have idle volumes.
- Idle volume. A volume with no I/O activity for at least six minutes.

The cluster automatically wakes one or more nodes to carry out requests when needed and eventually revives all nodes if required. If there is no intervening activity after the configured power.wakeAfter period, the cluster wakes all nodes to perform data, disk, and replication integrity checks.

Power Management Settings

Adjust these settings to best suit your Swarm implementation:

power.savingMode	Enables Power Saving Mode, which allows the system to go to sleep or power cap. Set to `False` to disable, which is called Full Performance Mode. A cluster with long periods of inactivity on nights and weekends can achieve significant power savings using this feature. Because only inactive nodes are affected, maximum available throughput is not affected, although additional latency is incurred on the first access to a node. Best practice Use Full Performance Mode (`power.savingMode = False`) for these situations: • The cluster is in constant use (24x7). • Uninterrupted feed restarts are critical for your operations. • The cluster is used for direct replication (DR).
power.sleepAfter	In seconds, 60 or greater; defaults to 2 hours. In Power Saving mode, how long a node is inactive before it becomes idle. This option determines how long a period of inactivity should occur with no incoming SCSP requests before Swarm pauses the node's health processor and displays it as Idle in the Swarm UI or Console. If you select Full Performance Mode, a node never displays as Idle. Setting the value to a small number allows nodes to become idle after a reasonable period of inactivity (two hours by default). Important Avoid setting this value to a long period, which prevents Swarm nodes from becoming idle and taking advantage of power savings.
power.wakeAfter	In seconds; defaults to 8 hours. In Power Saving mode, how long a node is idle before it becomes active again. This option determines how long a node remains idle before Swarm wakes up an idle node to allow the health processor to validate disk content integrity and replicas. Setting the number to a small number reduces power savings, although the volumes and nodes return to an Idle state if additional client activity is not received. Important Avoid setting this value too close to the power.sleepAfter value, which would make the disks cycle quickly between sleeping and waking and reduce power savings. Avoid setting this value to a long period, which can put content at risk because the health processor cannot run on sleeping nodes. See health.startDelay for more about the health processor.

node.archiveMode	Disabled by default, which is the normal operating state. Set to TRUE to put the node into archive mode, where it remains idle in low-power mode without participating in cluster activity until its capacity is needed. Use this to provision new nodes into the cluster proactively, well before they are needed. Archive Mode lets you designate a new or empty node as an archive node so that it remains idle in low-power mode without participating in cluster activity until its capacity is needed. This lets you keep additional storage capacity online and available without paying for the associated power costs of the additional nodes. Once an archive node is activated by diminishing capacity in the remaining cluster, it attempts to aggressively fill itself to capacity with incoming write requests. Once its storage capacity is full, the node returns to an idle state until its stored content needs to be read.

Configuring the Overlay Index

- Requirements
- Determining status

Swarm has an Overlay Index, which provides increased scalability in a storage cluster. When enabled, the Overlay Index tracks object locations in a shared cluster-wide index, minimizing multicast traffic in the cluster network.

The Overlay Index creates a dynamic master index in RAM from the local node indexes of all existing objects in the cluster. When an SCSP request for an existing object is sent to a Swarm storage cluster, the Overlay Index locates the nodes containing the targeted object and directs the request to an appropriate node without multicasting to locate it. This process minimizes the multicast traffic and associated processing in your storage cluster.

The knowledge of existing nodes in the cluster is refreshed regularly and the Overlay Index evaluates whether changes to its model of the cluster are required every 5 minutes, ensuring the Overlay Index will be optimally spread across all available nodes in the cluster as new nodes are added or existing nodes are removed from the cluster. Adjustments to the Overlay Index's distribution due to changes in the nodes participating in the Overlay Index (particularly immediately after a cluster reboot) happen quickly but are not instantaneous. Distribution adjustments should not affect your client activity, however.

By default, the Overlay Index is enabled for clusters with 32 or more nodes. You can reconfigure the Overlay Index without rebooting the cluster by setting its SNMP OIDs:

Overlay Settings	Default	SNMP OID	Description
index.optimize404	TRUE	overlayOptimize404	Optional. Enables the Optimize 404 feature in the overlay index, which returns 404 without multicast where possible. With the Overlay Index enabled and fully populated and Optimize 404 enabled, Swarm provides faster 404 (Not Found) responses for missing objects.
index.overlayEnabled	TRUE	overlayIndexEnabled	Enables the overlay index.

index.ovMinNodes	32	overlayMinNodes	Count. The minimum number of cluster nodes needed to activate use of the overlay index. Lowering the number of nodes may affect throughput performance. Caution A value lower than the default may hurt throughput performance for small-object reads.

Requirements

Before implementing the Overlay Index, review its features and considerations:

Features	Considerations
Enables the Health Processor to more quickly discover and remove over-replication generated by failed volume recovery (FVR) and erasure coding recovery (ECR), which maximizes the disk space on your cluster nodes.	Requires additional RAM index slots to hold the additional index information. If your current node indexes are full, you must add additional RAM and reboot the nodes in order to take advantage of the Overlay Index.
Reduces the multicast traffic in your storage cluster network.	Due to the extra processing required to maintain the index, does not provide scale benefit in smaller clusters and may negatively affect read throughput performance if the minimum number of nodes is lowered below the default of 32 nodes.
Provides faster 404: Not Found response time for nonexistent immutable and aliased objects when Optimize 404 is enabled.	

The Overlay Index requires additional index slots in cluster node RAM to store the additional index information. If there are not enough nodes or RAM resources to hold the additional index information, the Overlay Index will not be populated and the cluster will continue to use multicast to locate objects.

> See the Hardware Requirements for Storage for RAM requirements.

Determining status

To determine the current state of the Overlay Index at a given time, check the value of the indexOverlayStatus SNMP OID. The status of the Overlay Index will be one of the following:

- Disabled: When the overlay is turned off.
- Uninitialized: When the overlay is first activated.
- Operational: When population of the Overlay Index is in process.
- Authoritative: When the node servicing the SNMP query has determined that the Overlay Index is fully populated for the entire cluster. The Optimize 404 feature will only work when the Overlay Index is authoritative.

During normal operation, the state of the Overlay Index will toggle between "authoritative" and "operational" as the structure of the cluster changes with new nodes being added or existing nodes being removed. The status may also indicate a variety of reasons why the Overlay Index is not operational, such as insufficient nodes or inadequate Overlay Index space.

Configuring External Logging

- Obscuring UUIDs in Logs
- Configuring the Logging Host

This section deals with setting up the [log] section of your configuration.

Obscuring UUIDs in Logs

The log.obscureUUIDs parameter lets you control whether entire UUIDs display in logs.
With log.obscureUUIDs = false (the default), the entire UUID displays as follows:

```
<183>2016-02-11T17:06:10.359Z SCSP DEBUG: REQUEST: GET 172.16.0.33
<None>/358a76f06ffe7e4128d5e6c467a2a96d?
  alias=true (request:3087553833379006269 connection:25579392)
172.16.0.32 14/11 20:08:37.065
```

With log.obscureUUIDs = true , 20 of the 32 characters of the UUID display, as follows:

```
<183>2016-02-11T17:06:10.359Z SCSP DEBUG: REQUEST: GET 172.16.0.33
<None>/358a76f06f...c467a2a96d?
  alias=true (request:3087553833379006269 connection:25579392)
172.16.0.32 14/11 20:08:37.065
```

Keeping the default values makes issues easier to troubleshoot. However, set log.obscureUUIDs = true if you are concerned about the security implications of displaying entire UUIDs in your logs.

Configuring the Logging Host

The log.host option enables Swarm log messages to be sent to a central syslog, rsyslog, or syslog-ng server. The examples here show the specific Swarm items that need to be added to the UNIX configuration files for syslog or syslog-ng. The actual configuration files will likely contain additional information for logging messages from other hosts and other logging facilities.

> See syslogd reference and syslog-ng reference .

To configure your Syslog server:

1. Set the syslog or syslog-ng daemon options to enable logging from a remote host.
   Edit /etc/sysconfig/syslog to the SYSLOGD_OPTIONS to the following:

```
SYSLOGD_OPTIONS="-r -m 0"
syslog-ng
```

2. Configure other logging options.
   This example shows a sample configuration with the standard syslog program. When editing the syslog.conf file , remember that facility.level must be followed by a tab character to separate it from the destination specification. (See the syslog.conf manual page.)

   ```
   # /etc/syslog.conf
   local6.* /var/log/castor.log
   syslog-ng
   ```

   This example shows a sample syslog-ng configuration. (See the syslog-ng.conf manual page.)
3. In the filters section, add the following:

   ```
   #CAStor filter
   filter f_castor {facility(local6);};
   ```

   If a filters section does not exist, add one after the source statement and before the destinations section.
4. In the destinations section, add the following:

   ```
   #CAStor destination
   destination d_castor {file("/var/log/castor.log");};
   ```

5. Add the following log statement:

   ```
   #Caringo-specific additions #
   log {source(s_net){ udp();}; filter(f_castor);
   destination(d_castor);};
   ```

   The statement `(source(s_net){ udp();};` is the default remote service source for syslog-ng. If you use a different remote source name, replace `(source(s_net){ udp();};` with the name you used.
6. Enter the following commands to restart the service with your changes:

```
sudo /etc/init.d/syslog-ng stop
sudo /etc/init.d/syslog-ng start
```

7. Set Swarm logging options in the node or cluster configuration file.
    - log.host. The fully qualified host name or IP address of your syslog or syslog-ng server.
    - log.port. The server port.
    - log.level. Set the log level:

Level	Meaning
0	No logging
50	Critical messages and announcements
40	Errors, critical messages, and announcements
30	Warnings, errors, critical messages, and announcements
20	Info, warnings, errors, critical messages, and announcements
10	Debug, info, warnings, errors, critical messages, and announcements (the most verbose log level)

Fully qualified parameters:

```
log.host = 192.168.0.100
log.port = 514
log.level = 40
```

Section/unqualified parameters:

```
[log]
host = 192.168.0.100
port = 514
level = 40
```

Configuring an Rsyslog Server

You can configure an rsyslog server running Red Hat® Enterprise Linux® or CentOS operating system to accept incoming syslog messages from Swarm.

> See the rsyslog man page or the rsyslog documentation.

To configure the syslog server:

1. Log in as a user with root privileges.
2. Execute the following command:

   ```
   vim /etc/rsyslog.conf
   ```

3. In the rsyslog.conf file, comment out the following lines to accept inbound UDP connections on port 514:

   ```
   $ModLoad imudp.so
   $UDPServerRun 514
   ```

4. Edit the file so the timestamp and IP address of incoming syslog messages appear.
5. Locate the following text:

   ```
   #### GLOBAL DIRECTIVES ####
   $ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
   ```

6. Change this text to the following:

   ```
   #### GLOBAL DIRECTIVES ####
   $template myFormat,"%fromhost-ip% %rawmsg%\n"
   $ActionFileDefaultTemplate myFormat
   ```

7. (Optional) Create a log file for each Caringo product by configuring a log file per logging facility:

   ```
   local5.* /var/log/caringo/cr.log
   local6.* /var/log/caringo/castor.log
   ```

8. (Optional) Create a log file based on any desired string in the log message using the :msg parameter.
   For example, to create a log file that only includes messages with the word "Trims", use this format:

```
:msg,contains,"Trims" /var/log/caringo/trims.log
```

The result would match the following messages:

```
2016-02-11T17:06:10.359Z 10.1.1.153 [21] debug : 00:51,602 HP
DEBUG: Trims decidable locally / trims needed: 0/0
2016-02-11T17:06:10.359Z 10.1.1.153 [21] debug : 00:52,484 HP
DEBUG: Trims decidable locally / trims needed: 0/0
```

9. Check iptables and Security-Enhanced Linux (SELinux) to verify that you are not blocking inbound port 514.
10. Restart the rsyslog process:

```
service rsyslog restart
```

Configuring an External Time Server

Precisely synchronized time is critical to the integral processes in the Swarm storage cluster, such as versioning, lifepoints, and updates. If the nodes in a storage cluster are not synchronized with each other, you may experience unexpected results.

Guidelines for Time Servers

Swarm requires extremely precise clock synchronization to prevent data loss. Follow these guidelines to ensure adequate synchronization:

- Use NTP servers: (Network Time Protocol) servers to synchronize the clock in each cluster node.
- Do not use OpenNTPD or SNTP: Swarm only supports the NTP protocol. The Open Network Time Protocol Daemon (OpenNTPD) and the Simple Network Time Protocol (SNTP) are not supported because these protocols do not implement high-accuracy timing methods required by Swarm.
- Use trusted NTP servers: Be sure to use trusted NTP servers, whether they are dedicated hardware solutions in your internal network or external, public servers.
- Synchronize client systems: Swarm does not synchronize the client system clocks. Best practice is to synchronize these clocks with the NTP servers as well. However, client system configuration is not required.
- No virtual deployments in production : Clock drift in virtual environments may prevent Swarm from being able to properly sequence updates; therefore, virtual deployments are not supported for production use, only for test and development clusters.

Configuring a Node with NTP

To configure a node to use NTP, implement the timeSource setting in the node or cluster configuration file by setting the network.timeSource setting to one or more IP addresses or host names.

> Important
> If the nodes do not have network access to public NTP servers, they will eventually time out waiting for a connection and automatically restart. Ensure that the timeSource setting is set correctly or that the nodes have network access to the NTP pool servers.

For best results, set the network.timeSource setting to one or more IP addresses or host names based on the DNS server configuration in the network.

If node is not conf igured to a DNS server	Set the dnsServers setting to a valid value and set the network.timeSource setting to one or more IP addresses: `network.timeSource = 192.168.0.20 192.168.0.50 192.168.0.110`
If node uses DHCP that provides a DNS server	Set the network.timeSource setting to one or more host names. For example, you may decide to use NTP pool servers. The NTP Project recommends using pool servers that are close to your servers' time zone as described on their Help page (to view the page in a language other than English, go to ntp.org a nd click "How do I use pool.ntp.org?")
If node uses U.S.-based pool servers	Use the following setting value: `network.timeSource = 0.us.pool.ntp.org 1.us.pool.ntp.org 2.us.pool.ntp.org`

Configuring a Node without NTP

Never run a storage cluster in a production environment without using NTP. However, in demonstration or development environments where there is no internal or external NTP server available, you can choose a minimum of one or a maximum of two nodes as the master clock and synchronize the clocks in the remaining nodes to the master clock in one of those nodes.

> Warning
> If you create a Swarm storage cluster without an external NTP time source, ensure that the BIOS clocks in all new nodes are set relatively close to the correct GMT time before they join the cluster. All Swarm node clocks

are set to GMT (not local time), and they do not change for daylight saving time.

To implement a cluster without using NTP:

- Set the following setting in the configuration file of any single node in the cluster:

```
network.timeSource = system
```

All other nodes in the cluster will attempt to synchronize their clocks to this node.

Configuring Encryption at Rest

About Encryption at Rest

Swarm gives you the option to encrypt all user data on drive volumes. Swarm encrypts the data as it writes it to the drive and decrypts it on access. Because this occurs down at the kernel level, the effect is invisible: there is no difference in how you access encrypted versus unencrypted objects. You control encryption entirely through `[disk]` settings in your configuration.

What it protects — Swarm volumes generally contain sensitive and proprietary customer information. Implementing encryption at rest gives you two types of protection:

- Security for the data on all removed and failed physical drives.
- The ability to render all data in a cluster inaccessible by purging the encryption key.

> Important
>
> Swarm data is only encrypted when "at rest" (stored physically on the Swarm drives). The following is not encrypted:
>
> - objects within Swarm memory
> - objects in network transit between Swarm volumes or clusters
> - object metadata that is sent to Elasticsearch for indexing

What it does — Encryption only exists at the drive level, never during transmission. This is how encryption at rest works:

- Data on each new drive is encrypted with an administrator-supplied key that Swarm accesses from the volume.cfg/cluster.cfg file (Swarm does not persist any copies of the keys within the cluster).

- When an encrypted drive is removed from a chassis, the data on it cannot be accessed anywhere, without the administrator-supplied key.
- When an encrypted drive is moved to another cluster, Swarm can access the data on it by using its administrator-supplied key.
- If the administrator-supplied key is destroyed, all data on encrypted Swarm volumes are permanently and safely inaccessible.
- If the administrator-supplied key is lost, there is no method for recovering that data.

> Caution
>
> Caringo cannot restore lost keys, and there is no master key. Swarm cluster administrators are solely responsible for managing these keys, and they must secure the key by storing a copy in a safe and secure location.

Key concepts— Only volumes that encrypt data make use of encryption keys, and Swarm never reformats existing Swarm volumes to enable encryption. Here are some implications:

- Existing volumes — If you enable encryption on an existing storage cluster, nothing changes until you add a new volume on a node that has the new settings. For Swarm to reformat a volume for encryption, the node must be rebooted with the new settings and the volume must be a non-Swarm (new or retired) drive.
- Mixed volumes — It is no trouble having a blend of encrypted and unencrypted volumes in your cluster and within a given node.
- Converting volumes — To have encryption implemented across your existing cluster, you will need to reboot the nodes with the new settings and systematically retire, reformat, and add back the volumes, at which time Swarm will set up the hardware for encryption. See Retiring Hardware.
- Replication — Because encryption is low-level (local to the drive), it has no effect on replication (or any other request), and the key is not needed by the target cluster. Each source volume, if encrypted, has the key needed to read its own objects and send them to the remote cluster. Whether replicated objects are encrypted is completely independent of how they were stored in the source.

Example — Suppose that you have three nodes in your existing source cluster and you decide to enable encryption at rest, so you update your configuration with the needed settings. After that, you add a fourth node (node4) to the cluster and boot it individually. Nodes 1 through 3 do not have the encryption settings because they have not been rebooted, but node4 does have the settings to apply to new volumes. Because you booted the node with fresh, non-Swarm drives, node4's volumes are all encrypted at rest. For each request node4 receives, it will use its encryption keys to write and read from its encrypted volumes. That is, if node3 needs to write an object to node4, node4 uses its keys to store the object with encryption automatically. Given a remote replication cluster, the principle is the same. Your source cluster (with node4) tells the target cluster to fetch specific objects for replication. node1 at the target cluster asks node4 at the source cluster for some objects. Since node4 has the keys, it can read the object from its own drive and return the object to node1 in the target cluster. How node1 in the target cluster stores the object is up to that node's configuration. Because encryption is completely local, the target cluster volumes are encrypted only if you set them up to be so.

Performance impact — Encryption while reading and writing is a CPU-intensive activity and you should typically expect to see a 10-30% performance overhead depending upon your workload and hardware. The 2010 Intel Core processor family and later include special AES-NI instructions that implement the more complex and performance intensive steps of AES encryption. These instructions where implemented by AMD in their processors starting late in 2011. Swarm's kernel takes advantage of the AES-NI instruction set if available in your CPU.

For more information, see Intel Advanced Encryption Standard Instructions and Wikipedia AES instruction set.

Tip
To determine if a given processor has AES-NI support, run `grep aes /proc/cpuinfo` from a Linux command shell.

Best Practices

- Approach encryption as a cluster-wide setting. Although you could enable encryption separately on each chassis (which has its own `volume.cfg`), use the same encryption settings cluster-wide.
    - Exception: You may use chassis-specific encryption keys, if needed.
- Provide physical security measures to ensure that your encryption keys are protected (on the physical chassis with USB sticks, or on their CSN) from both theft and loss.
- To ensure that your entire cluster is encrypted, update your configuration settings and then systematically retire and reformat your existing drives.
- To change your encryption key, systematically retire and reformat your existing drives so that they use the new key.
- To ensure that your Elasticsearch metadata content is also encrypted, Elasticsearch recommends that you implement the encryption method dm-crypt. However, Swarm does not automate or manage the encryption key process for Elasticsearch volumes.

Encryption Settings

Volume encryption settings and keys are configured through [disk] settings in the volume.cfg/cluster.cfg file.

[disk] Setting	Default	Description
encryptNewVolumes	`0 (False)`	Required. Boolean. The feature must be explicitly enabled. Indicates that new Swarm volumes should be encrypted with the specified primary key. If set to 1 (True), be sure to set `disk.encryptionKeyPrimary`.
encryptionKeys	`{}`	Required. A dictionary of key name/value associations. Swarm uses them to read volumes that were encrypted with these keys. Example: `{'cluster_key_5_15_2016': 'a24f8ec391ab3341', 'cluster_key_5_12_2015': 'de3498245ce8bf89'}` You use the `disk.encryptionPrimaryKey` setting to select which key in `disk.encryptionKeys` is “primary”. The other keys are considered to be “secondary”, and they are only used to read/write volumes that have already been encrypted.

encryptionKeyPrimary	*`empty string`*	String. Contains the name of the encryption key that is to be used to format new encrypted Swarm volumes. Required - If `disk.encryptionKeyPrimary` is set, `disk.encryptNewVolumes` must be True; if not (if it is blank), then `disk.encryptNewVolumes` must be False. - If `disk.encryptionKeyPrimary` is set, it must match one of the key names in `disk.encryptionKeys`, so copy the key name with care.
encryptionCipher	`aes-xts-plain64`	String. Indicates the preferred cipher algorithm to use when formatting new volumes. Keep the default unless directed otherwise. Valid values: aes-xts-plain64, aes-cbc-essiv
encryptionKeySize	`512`	Integer (in bits). Indicates the preferred key size to use when formatting new volumes. Keep the default unless directed otherwise. Valid values: 128, 256, 512
encryptionHash	`sha512`	Indicates the preferred hash algorithm to use when formatting new volumes. Keep the default unless directed otherwise. Valid values: sha256, sha512
encryptionIterationTime	`5000`	Integer (in milliseconds). Indicates the maximum iteration time that LUKS will use to derive a key to use when formatting new volumes. Keep the default unless directed otherwise. Valid values: positive integers ≥ 1000

Generating Encryption Keys

You may create your encryption keys using tools that are available from the standard Linux and Windows OS package repositories.

Here is an example script for generating a key:

Generate encryption key

```
#!/usr/bin/python

import random
random.seed()
print "%064x" % random.getrandbits(512)
```

Note that 64 hex digits is 256 bits, and 128 hex digits is 512 bits.

Enabling Encryption on a Cluster

When you enable encryption for your cluster and set up the required encryption keys, Swarm begins encrypting every new (not yet formatted by Swarm) volume that it detects and formats.

To implement encryption across your cluster, do the following:

1. Open your cluster configuration (`node.cfg/cluster.cfg` file) for editing.
2. Add appropriate values for these settings:

```
[disk]
encryptNewVolumes = true
encryptionKeys = {'key_2018-03-19': 'a24f8ec391ab3341',
'key_2016-09-27': 'de3498245ce8bf89'}
encryptionKeyPrimary = key_2018-03-19
```

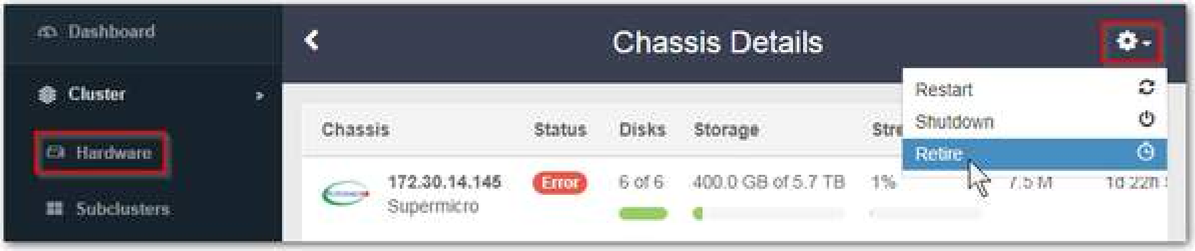
3. Important: Secure copies of your encryption keys.
4. Reboot the cluster to activate the settings change.
5. Add any new hardware, which Swarm will format for encryption.

> Important
>
> Any existing unencrypted Swarm volumes remain unencrypted, regardless of any hot-plugging you perform with them within the cluster. They will remain unencrypted and accessible without encryption keys. See next.

Encrypting Existing Swarm Volumes

To convert existing volumes to use encryption-at-rest, migrate the data using a chassis-by-chassis approach:

1. Enable the encryption-at-rest settings for the cluster, which will apply to newly formatted volumes.
2. Starting with the first chassis, retire it: from the Storage UI, select Cluster > Hardware, open the Chassis Details, and select Retire from the action (gear) menu.



3. Wait for all of its volumes to reach a status of "retired".
4. From the system console on the physical chassis, stop the storage processes: System Control > 3. Stop Storage Processes
5. Format all of the disks, which will now be encrypted: Disk Volumes > ALL
6. Reboot the chassis: System Control > 1. Reboot System
   As Swarm detects each new volume, it formats it as encrypted (using the `disk.encryptionKeyPrimary` value) and mounts it.

   > Note
   > You may change the key used to encrypt new Swarm volumes at any time, but your existing Swarm volumes will not be re-encrypted. To re-encrypt them, retire and reformat them, using the new encryption key.

7. With the first chassis complete, repeat all steps until every chassis has been converted.

Troubleshooting Encryption

Unmountable Volumes

This is how Swarm handles volumes it cannot mount:

Encrypted Swarm volume that Swarm cannot open	This occurs if the key is missing. Swarm ignores the volume and logs an error.
Encrypted non-Swarm volume	Swarm ignores the volume and does not format it.
Unencrypted non-Swarm volume	If non-blank storage volume that appears to have data from other operating systems is detected when a Swarm Storage node boots, a 60-second countdown timer is displayed on the physical console in order to give you time to prevent erroneous formatting. The most likely scenario for this happening is if you inadvertently PXE boot a server or laptop from the storage network. The countdown timer gives you the opportunity to power off the system before the drive is formatted. After the 60-second countdown timer expires, Swarm formats the volume.

Encryption Status and Logging

To see the encryption status of your volumes, you can view the SNMP volumes table for the cluster. You can also view status on the Chassis Details page of the Storage UI or the volume Status page of the legacy Admin Console: Swarm puts "(encrypted)" after the volume ID of volumes that are encrypted. If Swarm cannot open a volume because of a problem with the encryption key, Swarm writes a console message to that effect.

On startup or hotplug events, this is how Swarm logs encryption:

- If encryption is enabled (the `encryption.primaryKey` setting is set), each non-encrypted drive that is mounted is logged to the console, “Mounted non-encrypted volume /dev/sda”. An error is logged to syslog.
- If an encrypted volume cannot be mounted (such as for a missing key), an error is logged to the console, “Unable to mount encrypted volume /dev/sda”. An error is logged to syslog.
- When a volume is mounted, the log entry of that volume includes the encryption status of the volume.
- During hotplug events, a volume with a non-Swarm encrypted partition is mounted and formatted as a Swarm volume immediately.

Disabling Encryption on a Cluster

When you disable encryption on a cluster, the change affects how Swarm will format any new volumes it detects. Existing encrypted volumes, even if moved (hot plugged), remain encrypted and accessible only with their encryption keys.

To remove encryption entirely from your cluster, do the following:

1. Edit your cluster configuration (volume.cfg/cluster.cfg file):
   a. Disable new volume encryption.
   b. Remove the primary encryption key designation. (This makes it a secondary encryption key.)
2. Reboot all nodes that should have unencrypted volumes (to activate the settings change).
3. Systematically retire all of the encrypted volumes. Swarm relocates the data to other volumes.
4. Add back each volume.
   - When Swarm detects each new volume, it formats it as unencrypted and mounts it.
   - As new data fills up each volume, it is unencrypted and requires no key.

Decommissioning an Encrypted Cluster

To decommission an encrypted cluster, ensuring that none of the encrypted data is ever retrievable, do the following:

1. Delete the encryption keys from your cluster configuration (volume.cfg/cluster.cfg file).
2. Destroy all copies of the encryption keys.

   > Reminder
   >
   > `disk.encryptionKeys` can have more than one key value, and any of the key values can be used to open an encrypted Swarm volume.

3. Reboot the cluster.
   - Without the keys, Swarm cannot mount the volumes, so all are out of service.
4. Remove, reformat, and repurpose the volumes.

"To have encryption implemented across your existing cluster, you will need to reboot the nodes with the new settings and systematically retire, reformat, and add back the volumes, at which time Swarm will set up the hardware for encryption. See the retiring hardware section for more."

Managing Volumes

This section describes how to manage the volumes in your storage cluster nodes.

> Note
>
> In normal operations, managing Swarm volumes does not require administrative actions. Special cases can occur if a volume or a node has a problem or if you decide to perform hardware maintenance on a node.

- How Swarm Responds to Drive Changes
- Moving Volumes between Nodes
- Replacing Failed Drives
- Returning a Stale Volume to Service
- Retiring Volumes
- Retiring Hardware

How Swarm Responds to Drive Changes

> Note
>
> To support hot plugging, Swarm requires control of all of the volumes (`disk.volumes = all`).

- If the Health Processor is actively scanning a drive when it is removed, I/O errors are recorded in the log. These errors are expected and do not indicate a problem.
- When a drive is removed, volume recovery (FVR) and erasure coding recovery (ECR) are both triggered, which includes creating new replicas or erasure set segments for objects that were stored on that drive.
- If you insert that drive in the same node or in a different node in the cluster, both recovery processes stop. There is a temporary state of over-replication because the returned volume has replicas or segments that were already recreated elsewhere. In time, the excess replicas or segments are deleted.
- If you insert a non-Swarm drive in a node, Swarm recognizes, formats, and mounts the drive as a new volume.
- If you insert a Swarm-formatted drive, either into the same node or into a different node, it continues to function as a volume without loss of data.
- If you insert a Swarm-formatted drive that was previously retired, the volume remains retired. No manual configuration or intervention is required.
- Messages display in logs and in the Swarm Admin Console to indicate that a drive was inserted or removed.
- If you are inserting multiple drives into the same server chassis, you need to wait 2 minutes between drive insertions to ensure the new drives are evenly distributed across the multiple nodes that may be running on the chassis.

> Caution
>
> When adding or relocating volumes to a node, ensure that the node has enough RAM to handle them. If not, it might be unable to mount some of the volumes.

> Warning
>
> Do not move Swarm drives between disk array controller types after they have been formatted by Swarm. Each controller reports available drive space to Swarm that is matched with the controller. For example, many

controllers claim the last section of the drive, reducing the total available space. If you switch your drives with another controller, the new controller may claim additional drive space that is not reported to Swarm, so Swarm may try to write data to non-existing space, generating I/O errors.

Moving Volumes between Nodes

Physical volumes can be moved between nodes as necessary to address hardware failures or other constraints as determined by an administrator.

When a volume goes off-line because of a volume failure, node failure, or node shutdown, the remaining cluster nodes will immediately ensure that the correct number of replicas exists for all of the objects in the cluster. If a volume or node returns to the cluster during this procedure and prior to the 14-day time limit, the remaining nodes will recognize the volume has returned to service and cease their efforts to replicate the volume's data.

> Important
>
> When adding or relocating volumes to a node, ensure that the node has enough RAM to handle them. If not, it might be unable to mount some of the volumes.

> Warning
>
> Do not move Swarm drives between disk array controller types after they have been formatted by Swarm. Each controller reports available drive space to Swarm that is matched with the controller. For example, many controllers claim the last section of the drive, reducing the total available space. If you switch your drives with another controller, the new controller may claim additional drive space that is not reported to Swarm, so Swarm may try to write data to non-existing space, generating I/O errors.

Moving clusters

Volumes can also be moved to nodes that are in a different cluster. When this occurs, the objects on that volume become part of the new cluster and will be checked for the correct constraints within the context of the new cluster.

Replacing Failed Drives

Swarm volumes can be replaced after either an admin-initiated Retire (see Managing Chassis and Disks) or a Swarm-initiated failure resulting from I/O errors (see Retiring Volumes). After a volume is marked either Retired or Unav ailable, it can be replaced.

Administrators can insert a drive into a running node without restarting the server, provided that the server hardware supports this function and disk.volumes = all is configured. This feature (called hot plugging or hot swapping) lets you add storage capacity to a node at any time.

> See Hot Swapping and Plugging Drives.

Identifying the Drive

When a volume is marked unavailable or retired, its physical drive light turns on and stays on for one hour. When you need help identifying a failed or failing drive, use the drive light features of the UI:

- Swarm UI: Click through Cluster > Hardware to view the chassis, and enable the drive light. To flash the drive light for a specific drive, click the disk light toggle in its summary row. When you enable drive lights manually, they will remain lit until you turn them off.

See Managing Chassis and Drives.

- Legacy Admin Console: The Identify feature lets you identify a Retired volume that needs to be replaced. However, if the volume was marked Unavailable, use process of elimination: identify each of the working volumes in the chassis to determine which one does not flash and therefore needs to be replaced.

Once you have identified the correct drive, you can simply remove the drive and verify its serial number with the message in the UI. When you insert a new drive, Swarm will recognize that a new volume is available and will then format it for use.

> See Drive Identification Plugin.

Suspending Volume Recovery

While replacing a failed hard drive, be sure to suspend volume recovery:

- Swarm UI: In the Swarm UI, administrators can suspend an in-process volume recovery using the Suspend Recovery option under the settings (gear) icon. After the drive is replaced, resume the recovery using either the Enable Disk Recovery button in the banner message or the Enable Recovery under the settings gear icon.

  > Tip
  >
  > For drive-related events requiring user action (such as drive removal), Swarm helps you locate the hardware by including the SCSI locator (bus ID) and volume serial number in the log message that displays in the UI. (v9.2)

- Legacy Admin Console:
  1. In the Settings menu, select Volume: Suspend Recovery.
  2. Remove the defective drive and install the replacement drive.
  3. Ensure that the new drive appears in the Swarm Admin Console and has a non-zero stream count after several minutes of cluster activity.
  4. In the Settings menu, turn off Volume: Suspend Recovery.

Returning a Stale Volume to Service

The storage cluster is designed to automatically adapt when a volume (hard drive) or node fails for any reason. Swarm checks every storage cluster volume during the node startup procedure, and it tracks any gaps in service that would trigger a status change:

- If a volume is disconnected from the cluster for more than 2 weeks, it is considered "stale" and its contents cannot be used unless an administrator specifically overrides this process.
- If a node is shut down for more than 2 weeks, all of its volumes are considered stale and cannot be used.

The "stale" status is triggered by a service gap of 2 weeks, which is the default value for the disk.obsoleteTimeout setting.

You can force a volume remount by modifying the disk.volumes setting and adding the :k (keep) policy option. You can also return them to service dynamically (either remounting or reformatting) using SNMP. (v9.3)

Reformatting volumes (recommended)

Reformatting the volume allows it to be filled by the health processor in an orderly fashion. Doing so prevents creating excessive work for the health processor and prevents generating trapped space that will need to be reclaimed.

```
snmpset -v2c -c ourpwdofchoicehere -m
./CARINGO-MIB.txt:./CARINGO-CASTOR-MIB.txt
192.168.99.100 castorFormatStaleVolumeAction s "/dev/sda"
```

> Important
>
> The volume's encryption status is always retained on return to service; physical removal from Swarm is required to change it.

Remounting volumes

It is rarely desirable to remount a volume that has stale content. The volume's missing content will have been recovered by this time and so the cluster will have its full complement of replicas of the cluster's content. Adding extra replicas will create work for the health processor to sift through the replicas, cleaning up redundant and obsolete copies. This cleanup will create trapped space in the cluster that will take several HP cycles to reclaim.

> Note
>
> When you force a stale volume back into service, be aware that you could inadvertently resurrect content that was explicitly deleted by clients. This is not a problem for content automatically deleted by lifepoint policies because the obsolete content will be discovered and deleted by the Swarm health processor.

```
snmpset -v2c -c ourpwdofchoicehere -m
./CARINGO-MIB.txt:./CARINGO-CASTOR-MIB.txt
192.168.99.100 castorRemountStaleVolumeAction s "/dev/sda"
```

Retiring Volumes

- Triggers for Retire
- Canceling an Ongoing Retire

Due to current sophisticated disk storage devices and interfaces, the underlying disk system performs many error detection steps, bad sector re-mappings, and retry attempts. If a physical error propagates to the Swarm software level, there is little chance that a deterministic set of steps can be performed to work around the failure.

Additionally, there is no guarantee that the extent of the error can be isolated or that the continued use of a failing device will allow the node to continue operating normally with its peripheral storage devices. As a result, Swarm takes the conservative approach of retiring a volume when it receives a configurable number of I/O errors.

If the configured number of additional errors are received during the retire (disk.ioErrorTolerance), Swarm immediately marks the volume as Unavailable and kicks off both the volume recovery process (FVR) and the erasure coding recovery process (ECR) to relocate all the volume's objects.

Triggers for Retire

Swarm changes a volume's state to Retiring when any of these events occurs:

- You click Retire next to a volume on the node status page in the Swarm Admin Console.
- You click Retire Node, which retires all volumes on the node at the same time.
- The number of I/O errors specified by disk.ioErrorToRetire occur in the time period specified by disk.ioErrorWindow.

A Retiring volume accepts no new or updated objects. A volume remains in the Retiring state until all of the objects stored on that volume (including replicas) are moved to other volumes in the cluster. The Retiring state persists even if the node is rebooted. You may see the object count increase.

When all objects are moved, the volume state is changed to Retired and Swarm does not use the volume anymore. At that point, remove the volume for repair or discard it.

> Note
>
> If there are continued I/O errors that exceed the number specified by disk.ioErrorTolerance when the volume is in the Retiring state, the volume state is changed to Unavailable, regardless of whether Swarm has finished moving objects to other volumes.

Canceling an Ongoing Retire

You can cancel an ongoing retire by using the castorCancelVolumeRetire SNMP action. It takes a string to name a specific volume, or all.

Canceling retire on a specific volume

```
snmpset -v2c -c ourpwdofchoicehere -m
./CARINGO-MIB.txt:./CARINGO-CASTOR-MIB.txt
 192.168.99.100 castorCancelVolumeRetire s "/dev/sda"
```

Canceling retire on all volumes

```
snmpset -v2c -c ourpwdofchoicehere -m
./CARINGO-MIB.txt:./CARINGO-CASTOR-MIB.txt
 192.168.99.100 castorCancelVolumeRetire s "all"
```

Retiring Hardware

- Retire a Chassis
- Retire an Enclosure

These are typical reasons to retire hardware:

- Bad drives — Retiring volumes that are throwing errors or whose performance has dropped. Such volumes should not be returned to service in the cluster. (See Retiring Volumes.)

- Encryption-at-rest — Moving content off chassis so that its volumes can be reformatted to support encryption-at-rest. (See Configuring Encryption at Rest.)
- Planned EOL and upgrades — Replacing old, serviceable hardware with new equipment (a hardware refresh). Planned end-of-life for enclosures involves decommissioning the enclosure by retiring all of its chassis and moving the reformatted drives back into service.

Why not just hotplug? Although Swarm does support moving drives within a cluster for quick data migration, moving an entire set of drives from a server chassis or rack enclosure temporarily risks data loss because the hardware receiving the drives may hold several of the replicas/segments of the same object. In addition, moving drives requires a level of hardware compatibility, and some hardware situations will not support such drive moves:

- Use of incompatible RAID cards/tagging between chassis (especially true for those who emulate JBOD with single disk RAID-0 definitions).
- Inability of the controller in the new chassis/enclosure to recognize or otherwise work with the drives being moved (such as drive firmware vs. controller firmware, even with a pure HBA/JBOD setup).
- Inability of either the old and/or new equipment to properly support hot plug for drives.

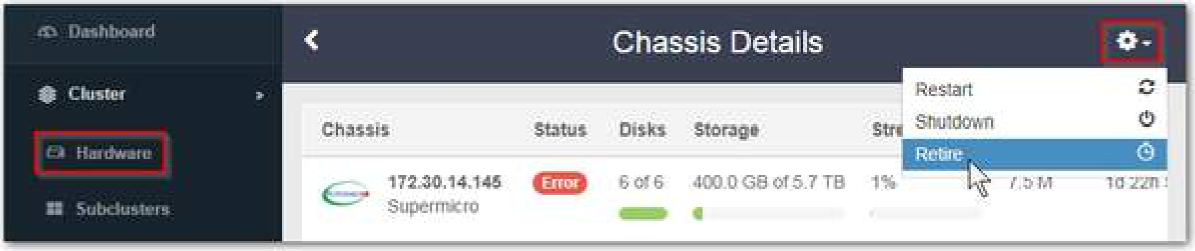
Best practice

The safest way to move all of the data being stored in an end-of-life chassis or enclosure is to retire the chassis and then format and reintroduce the drives.

Retire a Chassis

When you have Swarm retire a chassis, it retires all of the drives within it. If the drives are in good shape, they can be reformatted and returned to service.

1. From the Storage UI, select Cluster > Hardware, open the Chassis Details, and select Retire from the action (gear) menu.



2. Wait for all of its volumes to reach a status of "retired".
3. From the system console on the physical chassis, stop the storage processes: System Control > 3. Stop Storage Processes
4. Format any of the disks that might be returned to service: Disk Volumes > ALL
5. Shut down the node: System Control > 2. Shutdown System
6. Transfer reformatted drives to other chassis as appropriate. (See Hot Swapping and Plugging Drives.)

Retire an Enclosure

1. Add the new equipment to the cluster.
2. Retire each chassis in the old enclosure, following the procedure above to reclaim any serviceable drives.
3. When every chassis is retired (with drives reformatted for reuse) and shut down, you can power down and

remove the enclosure.

4. When you insert each of the reformatted drives into a chassis, the drive returns to service storing Swarm data.

> Note
> If retiring isn't feasible for your deployment, contact Support to review your cluster layout and put together a plan for disk migration that works best for you.

Using SNMP with Swarm

The Swarm SNMP agent implementation allows you to monitor the health of cluster nodes, collect usage data, and control node actions. You can integrate your storage cluster into an enterprise SNMP monitoring infrastructure.

> SNMP version
> Swarm supports SNMP version 2 only.

> Note
> Certain configuration parameters are persisted and are best set by SNMP. See Persisted Settings.

SMNP MIBs

If you boot from a Platform Server, see the following MIBs located at `/usr/share/snmp/mibs`:

- CASTOR-MGR-MIB.txt. An aggregate MIB for all cluster nodes.
- CARINGO-CASTOR-MIB.txt. A standard Swarm hardware MIB provided with the Swarm SNMP agent

If you do not boot from a Platform Server, see the CARINGO-CASTOR-MIB.txt MIB located in the root directory of the Swarm software distribution.

Swarm allows you to access the standard hardware MIBs distributed with the Net-SNMP package. These MIBs provide hardware reporting for areas such as processor load, memory availability, and network bandwidth.

> For complete information on the available OIDs, see the Net-SNMP MIB documentation.

Managing Swarm Nodes

- SNMP Commands
- Shutdown Action for Nodes
- Retire Action for Nodes and Volumes
    - Single volumes
    - Entire node

SNMP Commands

Storage cluster nodes are controlled through the SNMP action commands. The following OIDs enable you to disable nodes and volumes with nodes from a storage cluster:

- castorShutdownAction. Disable nodes and volumes within nodes for servicing.

- castorRetireAction. Disable nodes and volumes within nodes for retirement.

Shutdown Action for Nodes

To gracefully shut down a Swarm node, the string shutdown is written to the castorShutdownAction OID. Similarly, writing the string reboot to this OID causes a Swarm node to reboot.

When a node receives a shutdown or reboot action, it initiates a graceful stop by unmounting all of its volumes and removing itself from the cluster. For a shutdown, the node is powered off if the hardware supports this action. For a reboot, the node will reboot to machine, re-read the node or cluster configuration files, and startup Swarm.

A graceful shutdown is required to perform a quick reboot. Performing an ungraceful shutdown forces the node to perform consistency checks on all its volumes before it can rejoin the cluster.

> Tip
> Before shutting down or rebooting a node, check the node status page or the SNMP castorErrTable OID for critical error messages. Any logged critical messages will be cleared upon reboot.

> Note
> If you are rebooting more than one node at a time but not the whole cluster, wait at least 10 seconds in between each node reboot. This pause ensures that each node can communicate its rebooting state to the rest of the cluster, so that other nodes do not initiate recovery for the rebooting node.

Retire Action for Nodes and Volumes

The Retire action is used to permanently remove a node or a volume within a node from the cluster. This action is intended for retiring legacy hardware or pre-emptively pushing content away from a volume with a history of I/O errors. Retired volumes and nodes are visible in the Swarm Admin Console until the cluster is rebooted.

> See Retiring Volumes.

> Note
> The Retire action may take an extended amount of time to complete and requires at least three health processor cycles.

Single volumes

When a volume is retired, all of its stored objects are moved to other nodes in the storage cluster. After you initiate a volume retirement, the volume becomes a read-only volume and no additional objects can be stored on it. After all of the objects are moved to other locations in the cluster, the volume is idled with no further read/write requests.

Each volume is given a unique name within its node – the device string from the vols line in the configuration file. To retire a volume, its name is written as a string to the castorRetireAction OID. The volume retirement process is initiated immediately upon receipt and the action cannot be aborted after it starts.

To manually retire a volume,

1. Open the Swarm UI (or legacy Admin Console).
2. Click the targeted chassis/node (IP address).
3. For the targeted disk/volume, select Retire.

Entire node

Retiring a node means all volumes on the node are retired at the same time. After all volumes in the node are retired and the node data is copied elsewhere in the cluster, the node is permanently out of service and will not respond to further requests.

To retire a node and all of its volumes, the all string is written to the castorRetireAction OID. The node retirement process is initiated immediately upon receipt and the action cannot be aborted after it starts.

> Warning
>
> Ensure that the cluster has enough free space and nodes to store the objects from the retiring volume. For subclusters, this applies to the subcluster where the retiring volume resides. If the number of nodes in the cluster or subcluster do not have enough space to store at least two replicas of all objects, the retiring node cannot complete the retirement process until you add additional nodes. The Retire action does not require that the configured default replicas (`policy.replicas default`) is maintained to complete retirement. If there are not enough nodes to maintain the minimum number of replicas, messages will be logged that sufficient replicas could not be created.

SNMP Tools and Monitoring Systems

> SNMP version
>
> Swarm supports SNMP version 2 only.

Any standard SNMP query tool and monitoring system can be used to interact with Swarm. The examples in this section use the open source Net-SNMP (formerly UCD-SNMP) package that is available for UNIX and Microsoft Windows® platforms. Before using most tools and monitoring packages, install the Swarm MIB definition file. Follow the instructions included with the tool or package for more information.

Open Source Tools

The following tools can be useful to monitor and manage Swarm. Caringo does not endorse the applicability nor the fitness of these products when used within any environment.

- Net-SNMP (net-snmp.sourceforge.net). Provides command-line tools for UNIX and Windows environments to send and receive SNMP requests.
- Nagios (nagios.org). Provides web-based monitoring system for UNIX environments that can monitor systems and send alerts through email and pager.
- Zenoss (zenoss.com). An SNMP-based system for IT monitoring and management.

SNMP Examples with Swarm

To prepare to use the examples in this section, complete the following:

1. Record the IP address of a storage cluster node. If the cluster is not in your subnet, record the SCSP Proxy.
   In the examples below, the node's IP address is `172.16.0.32`.
2. Run the command from the directory that contains CARINGO-CASTOR-MIB.txt.
   For example, copy CARINGO-CASTOR-MIB.txt from the root directory of the USB flash drive or distribution to a local directory.
3. Record the following passwords:
   - read-only-password. The password for the read-only user defined in the security.operators setting.
     Default: `public`
   - read-write-password. The password for the read-write user defined in the security.administrators setting.
     Default: `ourpwdofchoicehere`

> See Defining Swarm Admins and Users.

Change in snmpwalk

To protect cluster performance, the 7.2 release changed the snmpwalk of the whole CASTOR MIB to make it skip several large, detailed tables in SNMP groups. With this change, administrators must upgrade from CSN v6.5 to update the CSN reporter.

If you need data from those skipped tables, you can create a targeted snmpwalk request. The snmp.getnextskips setting directs top-level snmpwalk to skip the groups and tables under the following: clusterConfig, responseHistogramTable, hp, clusterdata, indexer, configVariableTable, castorFeeds, feedVolTable, performance, recoveryTable

SNMP walk (snmpwalk) of all the Swarm values on a node:

```
snmpwalk -v 2c -c read-only-password -m +./CARINGO-CASTOR-MIB.txt
172.16.0.32 caringo
```

Request for a specific SNMP variable from a Swarm node:

```
snmpget -v 2c -c read-only-password -m +./CARINGO-CASTOR-MIB.txt
172.16.0.32 reads
```

Set request to shut down a Swarm node:

```
snmpset -v 2c -c read-write-password -m +./CARINGO-CASTOR-MIB.txt
172.16.0.32
 castorShutdownAction s shutdown
CARINGO-CASTOR-MIB::castorShutdownAction = STRING: "shutdown"
```

Set request that changes the cluster's sleepAfter setting to 7260 seconds (121 minutes):

```
snmpset -v2c -c read-write-password -m +./CARINGO-CASTOR-MIB.txt
172.16.0.32
 sleepAfter i 7260
```

SNMP Action OIDs

The "action" OIDs in Swarm are the SNMP objects that affect the operation of a node or the cluster.

> Important
> To prevent conflicts for cluster-level parameters such as volumeRecoverySuspend, the action should only be written to a single node to allow updates to the persisted settings UUID from a single node.

castorFeedRestartAction	Restarts a feed on a node using SNMP. When you set the OID value to a specific feed value, the feed restarts on all nodes in the cluster. The castorFeedTable OID allows you to view the Swarm feed information for a specific node. Each entry indicates a feed running on the selected node. The Swarm Admin Console allows you to view the SNMP Repository Dump page, which provides node-specific information.
logHost	Sets the logging host for writing log messages. When a node is booted, it sets the logging host based on the loghost parameter. Additionally, you can redirect syslog messages to your workstation to debug an issue.
logLevel	Sets the logging level. When a node is booted, it sets the logging level based on the loglevel parameter. You can increase the logging level to debug an issue and then return the level it to its previous value when completed.
nodeLogLevel	Sets the logging level for a specific node in the cluster, overriding the boot configuration specified by the loglevel parameter as well as the cluster-wide logLevel object.
logForceAudit	Sets forced audit logging for all nodes in the cluster, independent of the overall log level.

castorRetireAction	Removes the contents of a drive volume or an entire node in an orderly fashion. Instead of removing drives, consider retiring drives to save content that may not be saved on another drive. The device name from the node configuration vols parameter or the all string is written to this OID. You can simultaneously retire volumes from multiple nodes in the cluster.
castorShutdownAction	Sets a graceful shutdown or reboot a node or an entire cluster. The supported values are: • shutdown. Shuts down this node only. • reboot. Reboots this node only. • clustershutdown. Shuts down all nodes in the cluster. • clusterreboot. Reboots all nodes in the cluster.
volumeRecoverySuspend	Suspends volume recovery and erasure coding recovery behavior in the cluster during an upgrade or a network outage.

Practical SNMP with Swarm

This section outlines practical approaches in using the built-in SNMP agent to monitor the health and operational aspects of a storage cluster.

> Tip
> Although you can set up a simple ICMP ping monitor of a Swarm node, using the SNMP variables gives you detailed indications of drive and capacity problems.

Health Monitoring

The following variables can be used to monitor the basic health of a Swarm node. The volume table will have n from 1 to the number of volumes.

- caringo.castor.castorState. Should equal "OK."
- caringo.castor.castorVolTable.volEntry.volState.n. Should equal "OK."
- caringo.castor.castorVolTable.volEntry.volErrors.n. Should be zero.

If the monitoring console receives timeouts when trying to read these variables, there is something wrong with the node. If the state values are anything other than "ok," the node or the drives are transitioning from their normal state.

	Node	Volume
Valid states	OK Retiring Retired	OK Retiring Retired Unavailable

Any non-zero value in the volume error count indicates that a hard error has surfaced from the hardware through the OS driver and to the Swarm process.

Capacity Monitoring

The following variables can be monitored and collected for capacity alerting and reporting. The volume table will have n from 1 to the number of volumes.

- caringo.castor.castorFreeSlots. Should be greater than zero.
- caringo.castor.castorVolTable.volEntry.volMaxMbytes.n
- caringo.castor.castorVolTable.volEntry.volFreeMbytes.n
- caringo.castor.castorVolTable.volEntry.volTrappedMbytes.n

The castorFreeSlots variable indicates how many more objects a node can hold before it exhausts its memory index. If this occurs, the node is unable to store additional objects until objects are deleted or moved to other cluster nodes (or more RAM is added to the node). The free slots indicate how much RAM is required per object.

> See the Memory Sizing Requirements for RAM effects on node storage.

To compute the amount of disk space that is available for writing content, add the values volFreeMbytes and volTrappe dMbytes .

(volFreeMbytes + volTrappedMbytes) / volMaxMbytes = % free space on a disk volume
volUsedMbytes / volMaxMbytes = % space used by current context

Tip
You can total these disk usage variables for all volumes in a node and all nodes in a cluster to produce capacity utilization reports.

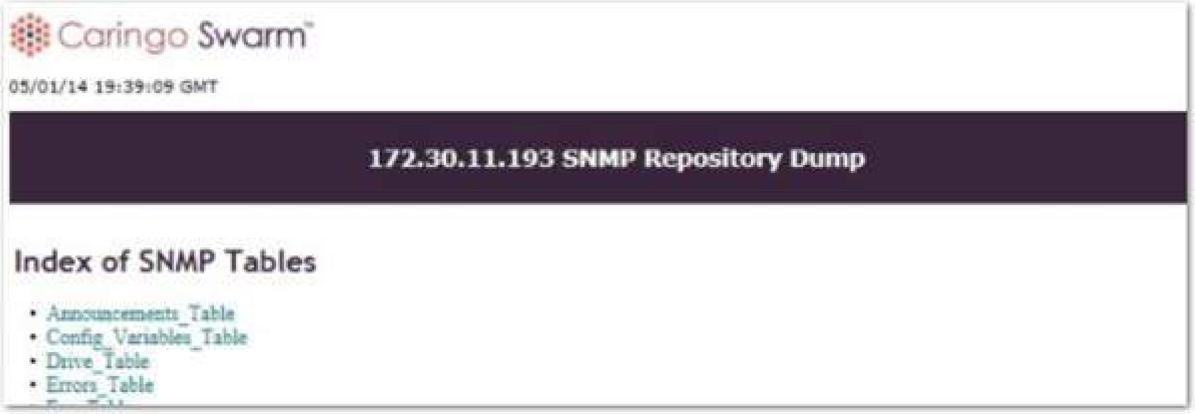
Client Activity Reporting

You can collect and report the amount of client activity received by the nodes to understand the end-user usage patterns and identify nodes that may be receiving significantly more activity than others. The resulting value can indicate a poor primary access node selection mechanism in the client application code.
The following SNMP variables indicate client request activity on a Swarm node.

- caringo.castor.scsp.writes
- caringo.castor.scsp.reads
- caringo.castor.scsp.infos
- caringo.castor.scsp.deletes
- caringo.castor.scsp.errors
- caringo.castor.scsp.updates
- caringo.castor.scsp.copies
- caringo.castor.scsp.appends

SNMP Repository Dump

The SNMP Repository Dump page provides node-specific information that is not available in the legacy Admin Console.
Example of the Volumes Table in the SNMP Repository Dump page:



05/01/14 19:39:09 GMT

172.30.11.193 SNMP Repository Dump

Index of SNMP Tables

- Announcements_Table
- Config_Variables_Table
- Drive_Table
- Errors_Table

Note that these SNMP items are no longer populated (v9.4):

- planarTemp
- tempStatus
- fanRedundancy
- psuRedundancy
- instantaneousWatts
- instantaneousMA
- minPowerCap
- maxPowerCap
- nics
- nicTable (including detail)
- nicFwVsn
- driveTable.driveStatus
- fans
- fanTable (including detail)
- psus
- psuTable (including detail)
- powerIntervals
- powerDrawTable (including detail)

If you rely on those SNMP values, they can be re-populated with a configuration change. Contact support for instructions.

To access the SNMP Repository Dump page for a cluster node:

1. Open the Swarm Admin Console.
2. In the Node IP column, click the IP address of the target node.
3. Scroll down and maximize Node Info .
4. Scroll down and click SNMP Repository .

> For more on the SNMP Repository Dump tables, see the SNMP MIB Reference file included in the top level of the Swarm product distribution ZIP file.

Statistics for Logical Usage

- Usage via SNMP and REST
- Usage via Metrics

Swarm calculates storage use by addressable object in order to support conventional (storage filer) reporting of file counts and their space usage. This approach tracks cluster-wide capacity by counting logical objects (the unique content of uploaded and versioned files) rather than actual streams (the raw space that is consumed by all Swarm components, including replicas, EC segments, context objects, and manifests). For example, if your cluster held just one 20 MB image with 3 replicas and 4 prior versions, only the versions would add to the object totals: 5 logical objects and ~100 MB logical space.

Swarm continuously sends its nodes updates about the cluster's logical usage (the current number of objects and the space they consume), which the nodes update with their own space-affecting activity. Swarm then aggregates these updates (for accuracy) and publishes them via SNMP and REST as `logicalObjects` and `logicalSpace`. A third statistic, `logicalUnprocessed`, exists to provide insight into the accuracy of the other statistics (the closer to zero it is, the more accurate they are). Swarm propagates this data quickly, so there is little lag behind the cluster activity that affects usage: writes, deletes, and updates. After a disk failure, however, you will see a drop in the aggregated estimates, followed by an increase to the true value, once Volume Recovery recreates the lost streams that were on that disk.

> Tip
> When you first boot your cluster after installing or upgrading to version 9.0, Swarm starts traversing the volumes to build these statistics, so they will not be accurate until that completes; however, the value of `logicalUnprocessed` indicates how it is progressing. Expect it to take 1 complete HP cycle to drop `logicalUnprocessed` to 0.

Usage via SNMP and REST

Swarm aggregates usage statistics from each volume and publishes them as cluster-wide values:

Published Aggregates	Units	Description	Accuracy

logicalObjects	count	The number of unique objects (including historical versions) stored for the entire cluster. Each content object counts only as 1, regardless of the number of replicas or EC segments that comprise it.	Approaches the actual number of logical objects in the cluster, minus context (domain, bucket) objects. Note Logical counts are estimates, and they are not accurate during volume recovery. The estimating is a consequence of Swarm's robust, no-single-point-of-failure design: Swarm keeps no master list of objects, so counts are inferred from multiple overlapping sources of information.
logicalSpace	MB	The logical space stored for the entire cluster, including historical versions (which are separate objects).	Includes both the data and the persisted headers on each object, with header newlines counting as two characters (‘\r\n’). EC encoded objects may include a small overage.
logicalUnprocessed	count	The number of streams in the cluster that have not been accounted for in `logicalObjects` and `logicalSpace`. After implementation, it will drop until it catches up, approaching zero.	When compared to the number of streams in the cluster, lets you roughly verify the other statistics, especially following the first boot after it is implemented.

> Remember
>
> These are cluster-level statistics, so each node is publishing the same values.

You can get `logicalObjects`, `logicalSpace`, and `logicalUnprocessed` by polling a node using SNMP:

SNMP for usage

```
snmpget -m +CARINGO-CASTOR-MIB -v2c -M +/usr/share/snmp/mib2c-data
-cPASSWORD -OQs
 {node- ip} logicalSpace
```

Alternatively, you can get `logicalObjects`, `logicalSpace`, and `logicalUnprocessed` by polling a node using the REST API:

REST call for usage

```
http://{node- ip}:91/api/storage/clusters/{clustername}
```

Trends - Each volume in a Swarm cluster is computing partial statistics for logical objects with replicas on other volumes. Swarm works to keep the correct number of replicas (and EC segments) for every object, but, if there are too many replicas, the statistics will trend higher. In the case of hardware failure, the statistics will trend lower while the recovery is taking place.

Timing - Each volume has accurate partial statistics immediately after a write or delete. REST API statistics are immediately available after each volume broadcasts messages that are sent every 30 seconds, but SNMP adds up to another 60 seconds for periodic polling of the aggregated values. Metrics does not aggregate, so the periodic metrics reports will be current with respect to the accounting cursor.

Usage via Metrics

Usage statistics are also reported via Swarm's Metrics mechanism. These metrics are checked on demand, at query time. Although Swarm publishes the statistics under the volume metrics, be aware that the values represent the cluster level:

Volume Metrics	Units	Description
logical_objects	count	The number of unique objects (including historical versions) stored for the entire cluster. Each content object counts only as 1, regardless of the number of replicas or EC segments that comprise it.
logical_space	bytes	The logical space stored for the entire cluster. The logical_space value is in bytes, not MB, for greater accuracy.
logical_unprocessed	count	The number of streams (replicas, EC segments, etc.) in the cluster that have not bee n counted for `logicalObjects` and `logicalSpace`.

Troubleshooting Storage

This section provides information to help you troubleshoot and resolve issues in your Swarm storage cluster.

- Troubleshooting Boot Errors
- Troubleshooting Configuration
- Operational Problems

Troubleshooting Boot Errors

Symptom	Action

When booting, the node generates an error stating that a boot device is not available. The node boots into an operating system other than Swarm.	If you are booting from a USB device, verify that the node is capable of booting from a USB device and the USB memory device is configured as the primary boot device. If you are PXE booting, ensure that: • The server is configured to network boot. • PortFast is configured on the switch ports that lead to the Swarm node. Otherwise, the extended time delay required for listening and learning Spanning Tree states can prevent netboot from delivering the Swarm image to the node in a timely manner.
The node boots from the USB device, but Swarm fails to start. The node begins to boot but reports a "kernel panic" error and stops.	These symptoms usually indicate a hardware compatibility issue with the hardware. Contact Swarm Support with the details of your hardware setup.

Troubleshooting Configuration

Symptom	Action
After the system boots, a message appears stating that the configuration file is missing. The node boots, but storage is not available on the node. A hard drive in a node does not appear as available storage. After adding a new hard drive to a node, some of the volumes will not mount. After moving a volume between nodes, some volumes in the new node will not mount.	Ensure that each node has a node.cfg file on the USB stick and that disk.volumes is properly specified. See Configuring the Nodes. The volume must be larger than the minimum value specified by the disk.minGB parameter (64 GB by default) or it will not boot. Small disks can be booted by lowering the size value in disk.minGB. If the vols specification is correct and the volume is larger than disk.minGB, this may be an issue with the amount of RAM in the node. Check the available RAM and ensure that it is provisioned sufficiently.
The node boots from the USB device but Swarm fails to start. The node begins to boot but reports a "kernel panic" error and stops.	These symptoms usually indicate a compatibility issue with the hardware. Contact Swarm Support with the details of your hardware setup.
Some changes to the node.cfg file disappear after editing.	If a USB flash drive is removed from a computer without unmounting, changes can be lost. Use the method for your OS to stop and unmount the USB media before removing it.

The following alert displays in the Swarm Admin Console: `Local clock is out of sync with node ip-address` The following indicator displays for each node where the error occurs: 🕒	A clock synchronization issue exists between the cluster nodes. The clock icon displays next to any node that sends a data packet that is more than three minutes offset from the reporting node's local clock. The local node will also log a critical error. Ensure that your Network Time Protocol (NTP) settings are correct and the cluster nodes can access the configured NTP server. See Configuring an External Time Server.
A node hangs during boot while initializing ACPI services.	System hardware is conflicting with the Advanced Configuration and Power Interface (ACPI) interface. To resolve this issue, add the argument acpi=off to the syslinux.cfg file on the USB flash drive (for local booting) or to the PXE configuration file (for network booting).
The Swarm node boots as having an unregistered license.	The license file is not in the Caringo directory on the USB drive, or the licenseFileURL option in the node or cluster configuration file is not set properly.
Inconsistent performance when using virtualized disks.	There may be timeout issues. Add the disk.deviceTimeout configuration parameter and increase the value as needed for your virtualization environment.

Operational Problems

For disk-related events requiring user action (such as disk removal), Swarm helps you locate the hardware by logging the SCSI locator (bus ID) and volume serial number at CRITICAL and ANNOUNCE log levels, which makes them display in the UI. (v9.2)

Helpful statistics

Swarm keeps statistics on incomplete read and write requests, which can help you diagnosis clients that may be behaving incorrectly.

- SNMP: `clientPrematureCloseRead, clientPrematureCloseWrite`
- UI: Drill into the health reports for chassis-level statistics. For the legacy Admin Console, statistics appear on a node's status page, under Node Operations: SCSP: Client premature close (read), SCSP: Client premature close (write)

Tip

For disk-related events requiring user action (such as disk removal), Swarm helps you locate the hardware by including the SCSI locator (bus ID) and volume serial number in the log message that displays in the UI. (v9.2)

Symptom	Action

A volume device failed.	Allow the node to continue running in a degraded state (lowered storage) OR Replace the volume at your earliest convenience. See Replacing Failed Drives.
A node failed.	If a node fails but the volume storage devices are functioning properly, you can repair the hardware and return it to service within 14 days. If a node is down for more than 14 days, all of its volumes are considered stale and cannot be used. After 14 days, you can force a volume to be remounted by modifying the volume specification and adding the `:k` (keep) policy option. See Managing Volumes.
In the Swarm Admin Console, all remaining cluster nodes are consistently or intermittently offline. If you view the Swarm Admin Console from another cluster node, the original node appears offline to the second node, and so on. Each node appears as its own island where no other nodes appear reachable.	If a new node cannot see the remaining nodes in the cluster, check the Swarm network configuration setting in each node (particularly the `group` parameter) to ensure that all nodes are configured as part of the same cluster and connected to the same subnet. If the network configuration appears to be correct, verify that IGMP Snooping is enabled on your network switch. If enabled, an IGMP querier must be enabled in the same network (broadcast domain). In multicast networks, this is normally enabled on the router leading to the storage cluster, which is usually the default gateway for the nodes. See IGMP Snooping.
You have read-only access to the Swarm Admin Console even though you are listed in `security.administrators.` You cannot view the Swarm Admin Console.	You added an operator (a read-only user) to `security.operators` but did not add your administrator user name and password to `security.operators` as well. As a result, you cannot access the Swarm Admin Console as an administrator. To resolve this issue, add all of your administrator users to the `security.operators` parameter in the node or cluster configuration file. See Defining Swarm Admins and Users.
The network does not connect to a node configured with multiple network interface controller (NIC) ports.	Ensure that the network cable is plugged into the correct NIC. Depending on the bus order and the order that the kernel drivers are loaded, the network ports may not match their external labeling.
A node automatically reboots.	If the node is plugged into a reliable power outlet and the hardware is functioning properly, this issue may indicate a software problem. The Swarm system includes a built-in failsafe that will reboot itself if something goes wrong. Contact Support for guidance.

A node is unresponsive to network requests.	Perform the following steps until the node responds to network requests. • Ensure that your client network settings are correct. • Ping the node. • Open the Swarm Admin Console on the node by entering its IP address in a browser window (`http://{ip-address}:90`). • Attach a keyboard to the failed node and press Ctrl-Alt-Delete to force a graceful shutdown. • Press the hardware reset button on the node or power cycle the node.
The cluster is using more data than expected.	Using Elasticsearch, enumerate the `CAStor-Application` field to determine how much data is being written by which application. Many Caringo applications use this metadata header, and having it indexed lets you analyze which application created which content.

A node is not performing as expected.	In the `castor.log`, view the node statistics, which include periodic logging of CPU utilization for each process:
	`2015-11-05 16:13:22,`
	` 898 NODE INFO: system utilization stats:`
	`  pid_cpusys: 0.06,`
	`  pid_cputot: 1.67,`
	`  pid_cpuusr: 1.61,`
	`  sys_contexts_rate: 5728.00,`
	`  sys_cpubusy: 0.91,`
	`  sys_cpubusy0: 0.37,`
	`  sys_cpubusy1: 1.46,`
	`  sys_cpuio: 0.02,`
	`  sys_cpuirq: 0.01,`
	`  sys_cpusys: 0.06,`
	`  sys_cpuusr: 0.82`

Managing and Optimizing Feeds

- Configuring Target Clusters
- Optimizing Replication Rate
- Deleting Search Data
- Feeds with Versioning

Configuring Target Clusters

Uneven filling — If you are concerned about uneven filling of the target (DR) cluster of your replication feed, use one of these configuration strategies:

- Run DR clusters in full performance mode by disabling Power-Saving Mode: `power.savingMode = false` (SNMP: powerSavingMode)
- Lower the setting that limits the difference in capacity between volumes, which defaults to 20%: `bidding.idleCost = 20` (SNMP: biddingIdleCost)
  - For pure DR clusters with no other traffic, set it to 5% to compensate for sleep cycles or feed definitions that favor particular nodes/volumes: `bidding.idleCost = 5`
  - For mixed-purpose clusters where remote replication causes uneven filling in the cluster, set it to 10%: `bidding.idleCost = 10`

Optimizing Replication Rate

You may need to adjust the rate at which replication occurs for these situations:

Need	Cause/Concern	Action
Speed up replication	Large volume of very small objects	Contact Support for specific settings adjustments.
Slow down replication	Low bandwidth and full cluster might trigger denial of service	In your networking routers/switches, enable their native rate-limiting features.

How Swarm parallelizes replication

The replication feed feature in Swarm seeks a high degree of parallelism in replicating objects between your source and target clusters. For each replication feed on each node, the replication feed is creating a batch of items to replicate. The size of the batch may be as large as 200 items, or smaller if a batch cannot be filled in 30 seconds. Once the batch of items has been filled sufficiently, it is sent to a node in the target cluster using long-running GET request that waits for the target cluster to replicate the items in the batch. When the batch of work has completed, the source cluster node can fill another batch. This mechanism creates a constant load of replication work for the target cluster. These GET/retrieve requests are relatively small and do not, in themselves, use much bandwidth.

Each node in the target cluster may be accepting work from multiple source cluster nodes from any number of source clusters and any number of feeds. Additionally, the source cluster might have a larger number of nodes than the target cluster. To prevent target cluster nodes from being overloaded, each node in the target cluster does two things. First, it delegates replication work to other nodes in the target cluster as a way of balancing the load. Second, each node limits the number of replications that can be done simultaneously, regardless of source. Swarm's defaults assume a moderate client load.

Precise means of throttling can be achieved using networking technologies, including QOS bandwidth limiting and the use of bandwidth-limiting forward proxies out of the target cluster or reverse proxies into the source cluster.

Deleting Search Data

The Elasticsearch index (database) of search data remains on disk after you delete the feed; if you no longer need it, you need to delete it manually.

To delete the search data, you need to reference the search index, which is the same as the name of your cluster:

Delete search data

```
curl -X DELETE "http://{ip- elasticsearch}:9200/{cluster- name}"
```

Note
Your Elasticsearch server manages additional indices related to Swarm cluster: Swarm Storage and Gateway store historical metric information in rolling indices. Deleting your search data will not affect your historical data.

Feeds with Versioning

If you are implementing Swarm Versioning in your cluster, note that it is supported for both types of feeds. By default, feeds process each object only twice, on creation and deletion. With versioning enabled, feeds push objects as frequently as needed to ensure that they stay current.

> For an introduction, see Object Versioning in Swarm Concepts.
> For implementation, see Implementing Versioning in the SCSP Reference.

Replication Feed

Because object versioning is based on domains and buckets, which are replicated between clusters, object versions will also be replicated between clusters. The replication feed processes historical object versions as well as current object versions.

Required
To use versioning with replications feeds, make sure to upgrade both the source and target clusters to the same version of Swarm before enabling versioning in both clusters.

Replication feeds replicate all versions, current and historical, to the remote cluster and allows the remote cluster to decide whether to keep the versions and how to integrate them into its version chain linkages.

Troubleshooting
If an object is versioned before the bucket/domain in the target cluster is updated to enable versioning, it is possible for the older version to not be replicated in the target cluster. Should this occur, use the SCSP SEND command to re-transmit older versions manually.

Search Feed

The search indices represent the current version of every object in the cluster. When a formerly obsolete named or aliased object becomes the current version again, the version number is based on an update time, provided on the object's metadata. This ensures that when Swarm decides a replica is the new current version, that fact will eventually be updated in search. Because different replicas of the same object version may transition to "current" at different times, it's possible different replicas will update the Elasticsearch record for an alias or named object more than one time. The latest such update wins. Swarm's update collision mechanisms prevent duplicate ES updates and minimize redundant updates.

Effect on search indices

- Two new indices represent all existing versions of aliases and named and their delete markers, with no information about which is the current version.
- When versioning is enabled or suspended, every new named or alias version creation results in a new versioned name or alias record.
- When specific versions are deleted, either by SCSP or by HP, the corresponding versioned record is removed.
- The primary key of these records is the version ID (Etag), which is unique to each version.
- Both alias and named object versions have a flag that indicates whether the version is a delete marker, and that information is captured in these indices.
- The search index schema upgrade for versioning does not require reindexing of your data.

Effect on searches

- Your existing search queries do not need to change.
- You can add the versions query argument to obtain all existing versions.
- Swarm returns versions in the order of the version chain, starting with the current version and ending with the original version. (When simultaneous updates occur, Swarm saves all updates but determines which position each occupies in the chain.)

Elasticsearch for Swarm

Elasticsearch provides Swarm the capability for metadata searching and historical metrics, and it furnishes the data needed to populate the Swarm Management UI.

This section covers implementing and maintaining Elasticsearch (ES) with Swarm.

> For details on using the query arguments for listing and search operations on object metadata, see Searc h Queries.

Terms of Use

Caringo will keep the SCSP searching API stable and will insulate applications from ES schema changes.

> Warning
> Caringo may modify, without notice, the schema of the information contained within Elasticsearch and of the mapping template. Such changes could affect your implementation if you directly interface with Elasticsearch or change the template included with the Caringo version of the Elasticsearch software.

Guidelines - If you have need to customize schemas, templates, or queries using Elasticsearch with Swarm, follow these guidelines:

1. Caringo does not support customized schemas. If you need schema changes for direct-to-ES projects, run a separate ES instance for that purpose.
2. Because Swarm upgrades may include schema changes, always plan to test and adjust any custom direct-to-ES queries you may use. Again, if you have extensive need for direct-to-ES operations, consider running a separate

ES instance.

3. Use a test environment to verify that an update works with your direct-to-ES queries.

- Snapshot and Restore Search Data
- Monitoring Elasticsearch
- Managing Elasticsearch
- Swarm Historical Metrics

Snapshot and Restore Search Data

- Configuring the Plugin
- Configure the S3 Repository
- Creating a Snapshot the S3 Repository
- Restoring from a Snapshot

> Note
> This technique makes use of and requires the Content Gateway with S3 enabled.

Swarm builds and maintains your search data (index) through your Search Feed, and it will regenerate the search index should it ever be lost. You can trigger this regeneration at any time by running the Refresh command for your search feed in the Swarm Storage UI (or legacy Admin Console, port 90). However, a complete refresh (which verifies all of the data) takes a long time, during which your listings are unavailable.

If you need to ensure that your listings are never offline, there is a way that you can take a snapshot of your index data so that you can restore it for instant disaster recovery, using an Elasticsearch plugin. Because the Gateway can function as an S3 Repository, you can leverage the AWS Cloud Plugin to get Snapshot and Restore capability for your search data (index). To snapshot is to back up your search data to a file system or S3 (Swarm); to restore is to place a snapshot back into production.

These are key reasons for using the AWS Cloud plugin:

- Search Index Restoration: If your Search cluster has problems and the search index is lost, you can restore a snapshot so that applications that depend on listings and collections are not interrupted.
- Usage Snapshot: The usage metering indices written are temporary. To preserve data being written since the last backup, you can set up frequent snapshots.
- Data Move: If you are making changes to your Search cluster, you can restore a snapshot to the new location to minimize disruption in services.

> Important
> Refresh the feed for the restored index, and allow time for Swarm to verify the index data. Until it completes, any objects that were created, changed, or deleted after the last snapshot may be missing or appear erroneously.

Configuring the Plugin

These are the values that you will need:

Elasticsearch to backup	`http://<elasticsearch-node>:9200`
Content UI (Portal)	`http://<domain>/_admin/portal/`
Domain	<domain> in destination Swarm storage cluster
Bucket	<bucket> within the `<domain>`
S3 Endpoint	`http(s)://<domain>:<S3-port>`
Token ID (access key)	UUID
S3 Secret Key	generated when token is created

1. In the Content UI (Portal) on the Swarm cluster that will store the Elasticsearch snapshots, create an S3 token.

   > Best practice
   > Although you can back up Elasticsearch to the same Swarm cluster that is using it, it is best to use a separate Swarm cluster.

   a. Create or select the domain.
   b. Open its Settings (gear icon) and select the Tokens tab.
   c. Create a token that includes an S3 key.
   d. Record both the access key (token ID) and the secret (S3) key:

2. On each node in your elasticsearch cluster, install the AWS Cloud Plugin:
   a. Log into the node via ssh the root user.
   b. Install the plugin:

   ```
   sudo bin/plugin install --batch cloud-aws
   ```

   c. Restart the elasticsearch service.
3. Configure an S3 repository using the token (see below).
4. Test the plugin, as shown below:
   a. Take a snapshot.
   b. Delete the search index (which will cause your listings to fail).
   c. Restore your snapshot using the manifest file.
   d. Verify that the listings are working again.

Configure the S3 Repository

In these examples, the Elasticsearch repository is using S3 to store the search data snapshots.

1. Create the S3 repository using a command like the following. The base_path can be empty, or set if this bucket will be the backup destination for multiple Elasticsearch clusters.
    - The endpoint with the bucket in the host must be accessible from every Elasticsearch node (to verify, run 'curl -i http://**essnapshots.**mydomain.example.com/'). This requires either explicit `/etc/host` entries or wildcard DNS for the domain. If any node fails to contact the endpoint, you must delete the repo with "`curl -XDELETE 'http://elasticsearch:9200/_snapshot/myRepo'`" and PUT it again.
    - These configuration values (`endpoint, access_key, secret_key`) can be stored in `elasticsearch.yml` instead of the JSON body (see the Elasticsearch docs for the config names).

```
curl -XPUT
  'http://elasticsearch:9200/_snapshot/myRepo'
  -d '{
      "type": "s3",
      "settings": {
        "bucket": "essnapshots",
        "region": null,
        "endpoint": "http://mydomain.example.com/",
        "protocol": "http",
        "base_path": "myswarmcluster",
        "access_key": "18f2423d738416f0e31b44fcf341ac1e",
        "secret_key":"BBgPFuLcO3T4d6gumaAxGalfuICcZkE3mK1iwKKs"
    }
}'
```

2. List information about the snapshot repository:

```
curl -XGET
  'http://elasticsearch:9200/_snapshot'
```

3. Verify that the repository was created successfully:

```
curl -XPOST
  'http://elasticsearch:9200/_snapshot/myRepo/_verify'
```

Creating a Snapshot the S3 Repository

1. Create a new snapshot into S3 repository, setting it to wait for completion:

```
curl -XPUT
  'http://elasticsearch:9200/_snapshot/myRepo/snapshot_20171031
    ?wait_for_completion=true'
```

   If needed, you can restrict the indices (such as to Search only, if Metrics backups are not needed). See Elasticsearch documentation for details on restricting indices.

2. Allow several hours for this to complete, especially for an initial snapshot of an Elasticsearch with a lot of large indices.

Restoring from a Snapshot

1. Best practice — Always test that you can restore a backup before you need it! For example, delete the search index in a test or staging environment to simulate a situation where you need to restore Elasticsearch data:

   ```
   curl -XDELETE 'http://elasticsearch:9200/_all'  # do not do this in
   Production!
   ```

2. In the Storage UI:
   1. Open Cluster > Feeds, open your Swarm search feed, and select Actions (gear icon) > Pause to prevent a new index from being created.
   2. Open Settings > Cluster, Metrics and temporarily disable Swarm metrics (`metrics.targets` set to nothing) to prevent those indices from being created during restore.
3. Restore the search index, renaming indices if they exist and are locked:

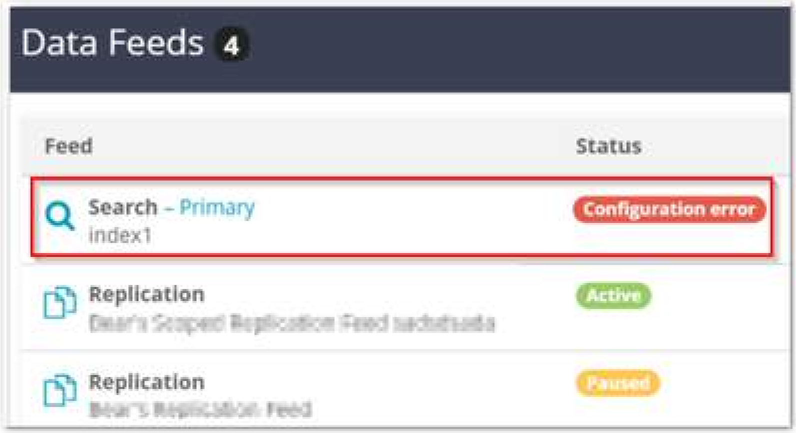
   ```
   curl -XPOST
   "elasticearch:9200/_snapshot/myRepo/snapshot_20171031/_restore" -d
   '{
     "rename_pattern": "(.+)",
     "rename_replacement": "restored_$1"
   }'
   ```

4. In the Storage UI:
   1. Open Settings > Cluster, Metrics and re-enable Swarm metrics (`metrics.targets` set to its prior value).
   2. Open Cluster > Feeds, open your Swarm search feed, and select Actions (gear icon) > Unpause to reactivate the feed.
5. Verify that your listings are working as before:

```
curl -iL 'http://swarm:80/?domains&format=json'
```

Monitoring Elasticsearch

Make it part of your routine to check the Swarm UI to monitor for problems with your Elasticsearch cluster. If you see that the Swarm search index status is yellow or red, examine the health of the Elasticsearch cluster.



Data Feeds 4

Feed	Status
Search – Primary index1	Configuration error
Replication Bear's Scoped Replication Feed sadsdsada	Active
Replication Bear's Replication Feed	Paused

Checking ES Cluster Health

A first step to any investigation is to query one of the Elasticsearch nodes for a report on the health of the Elasticsearch cluster:

1. If any Elasticsearch nodes are temporarily out of service for a known reason (such as a reboot or a rolling upgrade), wait until all nodes are back in service before proceeding.
2. Query the health of the cluster against one of the Elasticsearch nodes:

```
curl -X get <ES_Server>:9200/_cat/health?v
```

3. Verify that the value of node.total matches the expected number of nodes in the Elasticsearch cluster.

Diagnosing and Fixing Split Brain

A split brain situation is created when one or more nodes fails in a cluster and the cluster reforms itself with the available nodes. Believing the other clusters are dead, each cluster may simultaneously access the same data, which can lead to corruption.

1. Perform a health check of the ES cluster (see above).
2. If node.total is less than expected, perform the same health query against each of the other Elasticsearch nodes in the cluster.
3. If different Elasticsearch nodes report different values for node.total, then Elasticsearch cluster is experiencing

a split-brain situation.

4. Examine the /etc/elasticsearch/elasticsearch.yml configuration files and make sure that the Elasticsearch nodes are all configured correctly.
   If you need help confirming these settings, contact Support.
5. Examine the value of the "unassign" shards. If the value is greater than zero, there may be a shard allocation issue that is causing the Elasticsearch cluster to have a non-green status.
   Contact Support for help in resolving this situation.

Managing Elasticsearch

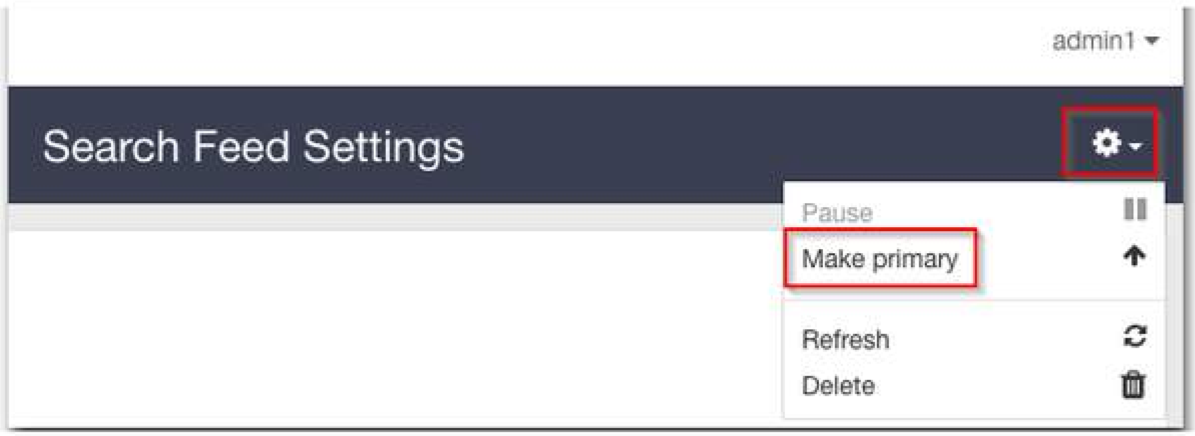
- Rebuilding a Search Feed
- Rolling Restart of Elasticsearch
- Upgrading Search and Metrics
- Merging and Renaming ES Clusters
- Resetting Elasticsearch
- Uninstalling Elasticsearch

Rebuilding a Search Feed

When the underlying schema for Swarm Search changes, new feeds are required to generate index data in the new format. Swarm Storage lets you create more than one Search feed so that you can transition from using one feed to another without disruption. During the transition, continue using the primary feed for queries; the second feed is incomplete until it fully clears its backlog. When the second feed is caught up, transition to it (marking it as primary) as soon as reasonable for your operations.

> Important
> When you verify that the new primary feed target is working, delete the original feed. Having two feeds is for temporary use only because every feed incurs cluster activity, even when paused.

1. In the Swarm UI, create a new search feed. Do not select Make primary.
2. Wait until the new feed has completed indexing the cluster, when the feed shows 0 "pending evaluation".
3. When the new feed is ready, make it the primary feed. In the Swarm UI, go to Cluster > Feeds, open the new Search feed, and select Make primary from the drop-down menu.



4. Operate with both feeds for several days. If there is a problem, you can restore the old feed to be primary during troubleshooting.
5. After this confirmation period, delete the old feed. In the Swarm UI, go to Cluster > Feeds, open the old Search feed, and select Delete from the drop-down menu.
6. If desired, delete the old index to reclaim that space.

Rolling Restart of Elasticsearch

Be aware that whenever any ES nodes go down, for whatever reason, Elasticsearch will immediately begin regenerating metadata (reallocating those shards). In controlled situations such as a rolling restart, you can take steps to minimize the impact.

A rolling restart of your ES cluster (keeping it operational while taking nodes offline one at a time) might be needed to upgrade the Elasticsearch version or to do maintenance on the server itself (such as an OS update or hardware change). Because Elasticsearch wants to keep data fully replicated and evenly balanced, it must be made to pause rebalancing until the rolling restart is done.

This pausing is done through ES settings changes, as recommended by Elasticsearch. The essential process is this:

1. Start maintenance mode by changing the ES settings.
2. Complete the maintenance work (such as upgrading ES) and rolling restart.
3. Stop maintenance mode by restoring the ES settings.

You should follow the steps in the Elasticsearch documentation, with this important addition:

1. When you first configured your Elasticsearch cluster, you set the `discovery.zen.minimum_master_nodes` value in the `/etc/elasticsearch/elasticsearch.yml` config file to be: (number of master-eligible-nodes/2, rounded down) + 1
   Verify that that number is smaller than the number of currently available nodes. If not, run the following command to set that number to be <current_min_master_nodes>: (current number of master-eligible-nodes/2, rounded down) + 1.

```
curl -s -XPUT 'http://ES_NODE:9200/_cluster/settings' \
 --data-binary '{"transient": {"discovery.zen.minimum_master_nodes"
: "<current_min_master_nodes>"}}'
```

2. Follow the Elasticsearch rolling restart procedure: www.elastic.co/guide/en/elasticsearch/guide/current/_rolling_restarts.html
3. After completing the rolling restart, reset `discovery.zen.minimum_master_nodes` to its original value, or adjust it based on the current number of expected available nodes:

```
curl -s -XPUT 'http://ES_NODE:9200/_cluster/settings' \
 --data-binary '{"transient": {"discovery.zen.minimum_master_nodes"
: "<original_min_master_nodes>"}}'
```

Upgrading Search and Metrics

To upgrade the Caringo RPMs for Swarm Search and Historical Metrics (but not Elasticsearch itself), do the following:

1. From Caringo Connect, download the Swarm Storage distribution (ZIP) to get the latest Caringo RPMs.

   ```
   caringo-elasticsearch-search-<version>.noarch.rpm
   caringo-elasticsearch-metrics-<version>.noarch.rpm
   ```

2. In the Swarm UI, pause your Search feed.
3. Initiate maintenance mode. See Rolling Restart of Elasticsearch.
4. On each ES server, install and configure the Elasticsearch components.
   a. If you have not yet added the Caringo RPM public key that is included with the distribution bundle to your system, do so now:

      ```
      rpm --import RPM-GPG-KEY
      ```

   b. Stop the Elasticsearch service:

      ```
      systemctl stop elasticsearch
      ```

   c. Upgrade Swarm Search:

      ```
      yum update caringo-elasticsearch-search-<version>.noarch.rpm
      ```

   d. On the system that runs the metrics curator, upgrade Swarm Metrics:

      ```
      yum update caringo-elasticsearch-metrics-<version>.noarch.rpm
      ```

   e. If directed by the release notes, make changes to your ES settings. See Configuring Elasticsearch.
5. With updating complete, start up Elasticsearch on each server. As root, start the Elasticsearch service:

   ```
   systemctl start elasticsearch
   ```

6. At this point, all ES servers are updated and started. Use one of these methods to verify that Elasticsearch is running (the status is yellow or green):

```
curl -XGET <ES·host>:9200/_cluster/health
```

```
systemctl status elasticsearch
```

7. Exit maintenance mode. See Rolling Restart of Elasticsearch.
8. In the Swarm UI, unpause your Search feed.

Merging and Renaming ES Clusters

This technique lets you merge two separate elasticsearch clusters or rename an elasticsearch cluster, which is done by retiring an Elasticsearch node "into" a new Elasticsearch cluster.

1. Join one or more existing nodes to the new cluster by changing their `cluster.name` value to the new cluster's name.
2. Verify that they joined.
3. To migrate shards off of the old nodes, you decommission a node by telling the cluster to exclude it from allocation.

   ```
   curl -XPUT localhost:9200/_cluster/settings -d '{
     "transient" :{
         "cluster.routing.allocation.exclude._ip" : "10.0.0.1"
      }
   }';echo
   ```

   This causes Elasticsearch to allocate the shards on that node to the remaining nodes, which is done without the state of the cluster changing to yellow or red (even if you have replication 0).
4. When all of the shards are reallocated, you can shut down the node.
5. To restore the node to service, include the node for allocation, which causes Elasticsearch to rebalance the shards again.

> See the Elasticsearch documentation.

Resetting Elasticsearch

If you should need to clear the state of Elasticsearch and Historical Metrics (such as if an index is deleted and erroneously recreated without the Swarm schema), you can perform a reset. A reset is a way to delete an index and refresh a feed safely, by deleting feeds before removing index data.

1. Set Swarm's configuration setting `metrics.target` to blank (which you can do via SNMP or REST).

```
curl -i -u admin:<PASSWORD> -XPUT --data-binary
'{"metrics.target":""}' \
 http://<SWARM·NODE>:91/api/storage/clusters/<CLUSTER·NAME>/setting
s/metrics.target
```

2. In the Swarm UI, delete the current Search Feed definition.
3. Delete the Swarm search index.
   a. Use the following command to determine the Swarm search index:

   ```
   curl http://<ES·NODE>:9200/_cat/indices | grep
   'index_<SWARM·CLUSTER·NAME>'
   ```

   b. Run the following command to delete the Swarm search index:

   ```
   curl -X DELETE http://<ES·NODE>:9200/<SWARM·SEARCH·INDEX>
   ```

4. Delete any or all of the historical metrics indices as needed.
   The following command deletes all of the metrics indices. Use care with the glob pattern, to avoid deleting indices that you want to retain.

   ```
   curl -X DELETE
   'http://<ES·NODE>:9200/metrics-<SWARM·CLUSTER·NAME>-*'
   ```

5. In the UI (Swarm UI or legacy Admin Console), create a new Search Feed definition that points to the ES servers. (This step creates the feed using the Swarm schema.)
6. Reinitialize the curator:

   ```
   /usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n -v
   ```

7. Reset the Swarm setting `metrics.target` back to its correct value (which you can do via SNMP or REST).

```
curl -i -u admin:<PASSWORD> -XPUT --data-binary
'{"metrics.target":"<METRICS-HOST>"}' \
 http://<SWARM·NODE>:91/api/storage/clusters/<CLUSTER·NAME>/settings/metrics.target
```

Uninstalling Elasticsearch

If you need to uninstall Elasticsearch for any reason, pause or delete the search feed before stopping the Elasticsearch service.

To uninstall the Search service:

1. Pause or delete the search feed. (See Viewing and Editing Feeds.)
2. Log in as root on the Elasticsearch server.
3. Stop the Elasticsearch service and uninstall the service:

   ```
   yum remove caringo-elasticsearch-search
   yum remove elasticsearch
   ```

   This does not erase the packages, so you can easily restore them later, if needed.
4. The Metrics curator relies on the Elasticsearch service, so do one of the following:
   - Reconfigure Swarm Metrics (`/etc/caringo-elasticsearch-metrics/metrics.cfg`) to point to a different ES cluster (or remove `cluster = <cluster-name>`)
   - Uninstall Swarm Metrics

     ```
     yum remove caringo-elasticsearch-metrics
     yum remove elasticsearch-curator
     ```

Swarm Historical Metrics

Although you have full access to instantaneous metrics on your storage cluster through SNMP, that route requires you to manage the sampling, recording, and querying of your historical data. The Swarm service for historical metrics, added in 8.1, gives you an easier way to collect the operational metrics and historical time-series data that are so valuable for your administration, billing, and planning activities.

The Swarm Historical Metrics Service uses an Elasticsearch cluster as a data repository, and it samples and records Swarm metrics autonomously at set intervals. You can access these metrics by querying Elasticsearch with aggregating (faceted) queries.

> Tip
>
> You can centralize your metrics management: Multiple Swarm clusters can use the same Elasticsearch

instance to collect their metrics.

> See Installing Swarm Metrics.

Components of Historical Metrics

Metrics Curator - The Metrics Curator is the service that does the work of creating and managing your metrics data. It deletes old metrics data, rotates the aliases, and creates new indices. You run the Metrics Curator manually only once, after installation and configuration (to prime the aliases); it is automatic after that, running daily at midnight (GMT) as a `cron` job.

Note
Only one Metrics Curator service should be running, so install it on one of your Elasticsearch nodes or another server running RHEL/CentOS 7.

Metrics Data - Swarm generates these metrics of interest:

1. feeds
2. index
3. node
4. volume (which includes usage statistics)
5. health
6. memory
7. scsp

These metrics are captured into one Elasticsearch index per cluster, per metric, per day (which means there are 7 × 7, or 49, indexes), with this naming pattern:

```
metrics-{cluster}-{metric}-{date}
metrics-clusterx-health-2018.03.18
```

Aliases combine multiple days of indices together into these useful collections:

```
metrics-{cluster}-{metric}-{alias}
metrics-clusterx-health-all
metrics-clusterx-health-today
metrics-clusterx-health-yesterday
metrics-clusterx-health-this_week
metrics-clusterx-health-last_week
metrics-clusterx-health-this_month
metrics-clusterx-health-last_month
```

Metrics Templates - To support historical metrics reporting, Swarm provides a custom set of Elasticsearch template schemas, which are located here:

`/usr/share/caringo-elasticsearch-metrics/bin`

These template files define the schema of the historical metrics from Swarm:

```
feedsschema.py
healthschema.py
indexschema.py
memoryschema.py
nodeschema.py
scspschema.py
volumeschema.py
```

- Resetting Swarm Metrics
- Swarm Metrics Troubleshooting

Resetting Swarm Metrics

- Reset Metrics Schema
- Reset Metrics Data

> Note
> To reset your entire Elasticsearch cluster, see Resetting Elasticsearch.

Reset Metrics Schema

To reset your metrics after a Swarm update that has a metrics schema change, you need to prime Swarm to restart with the new schema:

1. Stop the curator from running.
    a. Edit the configuration file for the metrics curator: `/etc/caringo-elasticsearch-metrics/metrics.cfg`
    b. Blank out the clusters value (which stops the metrics curator from processing any data), and save.
    c. (optional) To start with clean logs, delete the existing ones for the metrics curator:

    ```
    rm -rf /var/log/caringo/metrics_curator*
    ```

2. Allow Swarm to drop the old metrics schema.
    - Using the Swarm UI or SNMP, delete (set to blank) the value for `metricsTargetHost`.
3. (optional) To clear the prior metrics data, delete the old metrics data files.
    - For a specific cluster:

```
curl -X DELETE <ES·HOST>:9200/metrics-<CLUSTER·NAME>*
```

- For all clusters:

```
curl -X DELETE <ES·HOST>:9200/metrics-*
```

4. Reset and prime the curator:
   a. Edit the configuration file for the metrics curator: `/etc/caringo-elasticsearch-metrics/metrics.cfg`
   b. Set the clusters value to the original value (which enables the curator to process data), and save.
   c. Run the curator script manually to create the new indices and aliases:

```
/usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n
-v
```

5. Restart metrics in Swarm on the new schema.
   - Using the Swarm UI or SNMP, restore the value for `metricsTargetHost`.
6. To verify that metrics is working, check the Reports > History charts in the Swarm UI and the Usage charts in the Content UI.

Reset Metrics Data

To reset your Swarm metrics data only, such as when switching from test to production modes, use the Curator:

1. Edit the configuration file for the metrics curator: `/etc/caringo-elasticsearch-metrics/metrics.cfg`
2. Set retention to 0 (zero) days, commenting out the existing value, and save.
3. Run the curator script manually using the "-n" flag, so that it runs now:

```
/usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n -v
```

4. Edit the configuration file for the metrics curator and restore the original retention period.
5. Run the curator script manually again to create the new indices and aliases:

```
/usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n -v
```

Swarm Metrics Troubleshooting

Bypassing cluster validation

The Metrics Curator validates cluster names by requiring the Elasticsearch cluster to have an existing search index for the cluster of that name. Checking for this match enables the Curator to detect typos and other errors in configuring Swarm cluster names. However, there are two situations that require bypassing this validation:

- New cluster — You have a brand-new cluster for which no metadata has yet been indexed.
- No search — You have purposely omitted configuring a Search feed for your Swarm cluster, but you still want to enable Metrics.

To have the Curator skip this validation, add the -v (--valid) option to bypass the check. (v9.4)

Running curator manually

You can run the curator script at any time manually using the "-n" flag, so that it runs now:

```
/usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n
```

Verifying indices, aliases, and schema

Running the following cURL command lets you verify that the metrics indices and aliases exist. If they don't exist, then metrics curator may not have been run yet or may not be configured correctly.

```
curl "ESHOST:9200/metrics-CLUSTERNAME-feeds-all/?pretty=true"
```

If you need to reset your indices, see Resetting Swarm Metrics.

Running metrics on an old schema will cause problems. You can verify the version of your metrics schema by checking the properties:

```
curl "ESHOST:9200/metrics-CLUSTERNAME-feeds-all/?pretty=true"
...
"properties" : {
          "schemaversion" : {
            "type" : "string",
            "index" : "not_analyzed",
            "null_value" : "1.2"
          }
```

If you need to reset your schema after an upgrade, see Resetting Swarm Metrics.

"Data Unavailable"

When a metrics query succeeds with 200 OK but returns zero results, the Swarm Storage UI displays "Data Unavailable" for the affected usage chart. This is not an error, but it is meant to alert you that no data has been collected.

Each usage chart reports its own indices. For example, if the indices for feeds (`metrics-*-cluster-feeds-all`) have no results, then the Feeds chart will display "Data Unavailable".

To check whether the indices exist, run the following cURL command:

```
curl
"ESHOST:9200/metrics-CLUSTERNAME-scsp-all/metrics/_search?pretty=true"
```

No data being returned generally indicates one of the following situations:

- The storage cluster has not yet generated any metrics data (usually within 15 minutes of configuring metrics).
- Metrics is configured in Swarm Storage incorrectly, or not at all.

Indices in red status

If the Metrics Curator runs while one or more Elasticsearch servers are offline, the old metrics indices (which are deleted by Elasticsearch) could have a "red" status. That is because Elasticsearch will not be able to delete the shards of the index on the offline servers, and the status may remain red even after the all Elasticsearch servers come online.

If this happens, allow one day to pass for it to resolve itself. When the metrics Curator runs again on the following day, it should successfully delete the red metrics indices.

If the red status persists after two days, contact Caringo support for help correcting the situation.

Failed metrics query

If a query fails with a 400 Bad Request, a Swarm Metrics index was created with an invalid mapping.

1. Correct the problem in your settings.
2. Delete the bad index.
   The following command deletes all of the metrics indices. Use care with the glob pattern, to avoid deleting indices that you want to retain.

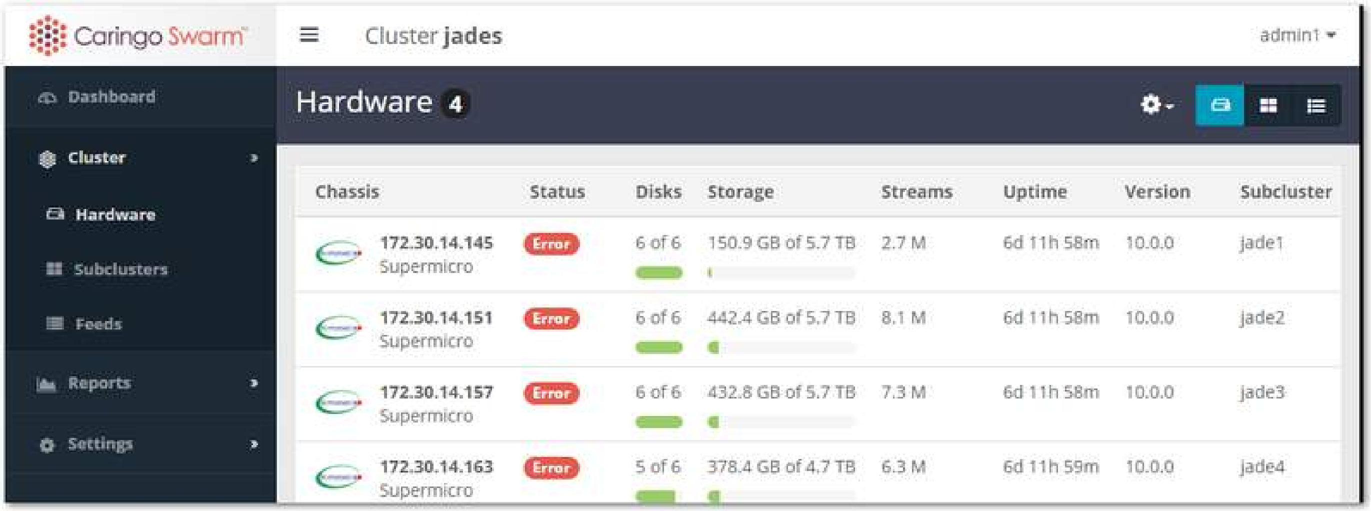
```
curl -X DELETE "http://ESHOST:9200/metrics-CLUSTERNAME-*"
```

3. Rerun the curator manually:

```
/usr/share/caringo-elasticsearch-metrics/bin/metrics_curator -n
```

Swarm Storage UI

The Swarm Storage UI (website) presents a comprehensive browser interface for monitoring and controlling your entire Swarm storage implementation.



Chassis	Status	Disks	Storage	Streams	Uptime	Version	Subcluster
172.30.14.145 Supermicro	Error	6 of 6	150.9 GB of 5.7 TB	2.7 M	6d 11h 58m	10.0.0	jade1
172.30.14.151 Supermicro	Error	6 of 6	442.4 GB of 5.7 TB	8.1 M	6d 11h 58m	10.0.0	jade2
172.30.14.157 Supermicro	Error	6 of 6	432.8 GB of 5.7 TB	7.3 M	6d 11h 58m	10.0.0	jade3
172.30.14.163 Supermicro	Error	5 of 6	378.4 GB of 4.7 TB	6.3 M	6d 11h 59m	10.0.0	jade4

The website offers your system and storage administrators a unified view of and easy access to the features and settings of Swarm:

- See all cluster chassis and drives, with both real-time and historical status and metrics
- Initiate cluster and chassis-level actions, such as restarting machines or retiring drives
- Create and manage search feeds, and define replication feeds, with optional filtering and SSL encryption
- View and change cluster settings dynamically
- Access event logs and advanced troubleshooting tools
- Identify drive volumes (using the drive light function)
- Monitor the health of the storage cluster, the Elasticsearch cluster, and all search and replication feeds

Accessing UIs

The functionality of the legacy CSN Console and Admin Console have been unified and replaced by the Storage UI.

Site	CSN	Platform
Swarm UI	`http://<CSN·host>:<cluster_admin·bindPort>/_admin/storage`	`http://<Platform-serve`
Content UI	`http://<Gateway·IP>:<SCSP·bindPort>/_admin/portal`	`http://<Gateway·IP>:<S`
Legacy Admin Console	`http://<CSN·host>:8090/services/storage`	`http://<storage·node>:`
Legacy CSN Console	`http://<CSN·host>:8090`	n/a

The `bindPort` refers to settings in the Gateway Configuration, and they offer the flexibility to support proxies and Docker environments.

> Deprecated
> The Legacy Admin Console (port 90) is still available, but it has been superseded by the Swarm UI. (v10.0)

- UI Essentials
- Viewing and Managing the Cluster
- Managing Chassis and Drives
- Viewing and Editing Feeds
- Using Cluster Settings
- Configuring SwarmNFS
- Using Cluster Reports
- Health Data to Support
- Legacy Admin Console (port 90)

UI Essentials

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Accessing the UI
- Navigating the UI
- Resources
- Rebranding the UI

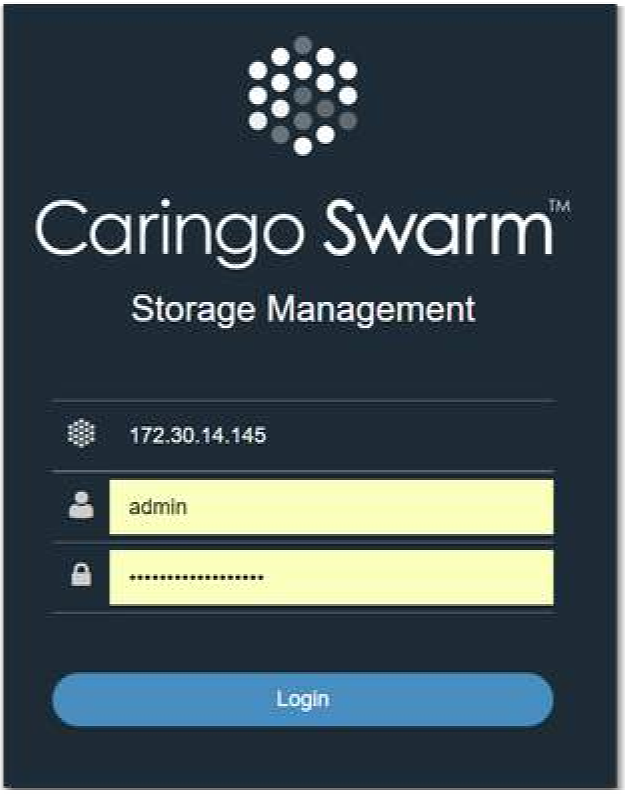
Accessing the UI

The URL used to access the Storage UI depends on where it is installed. See Installing the Storage UI to determine your starting URL.

> Tip
> Browsing to the Platform server (`http://{Platform·IP}`) will redirect you to the Swarm UI: `http://{Platform·IP}}/_admin/storage`

Once you've reached the UI, you will be prompted to login by specifying:

Host	Read-only. The host name or IP Address of the Swarm storage cluster to be viewed. Note The Storage UI connects to port 91. If the bind port was changed to anything other than 91, it must be specified: `{host}:{custom-port}`
Username	An admin username configured in the `security.administrators` parameter. See Defining Swarm Admins and Users.
Password	The password for the entered admin username.


Caringo Swarm™

Storage Management

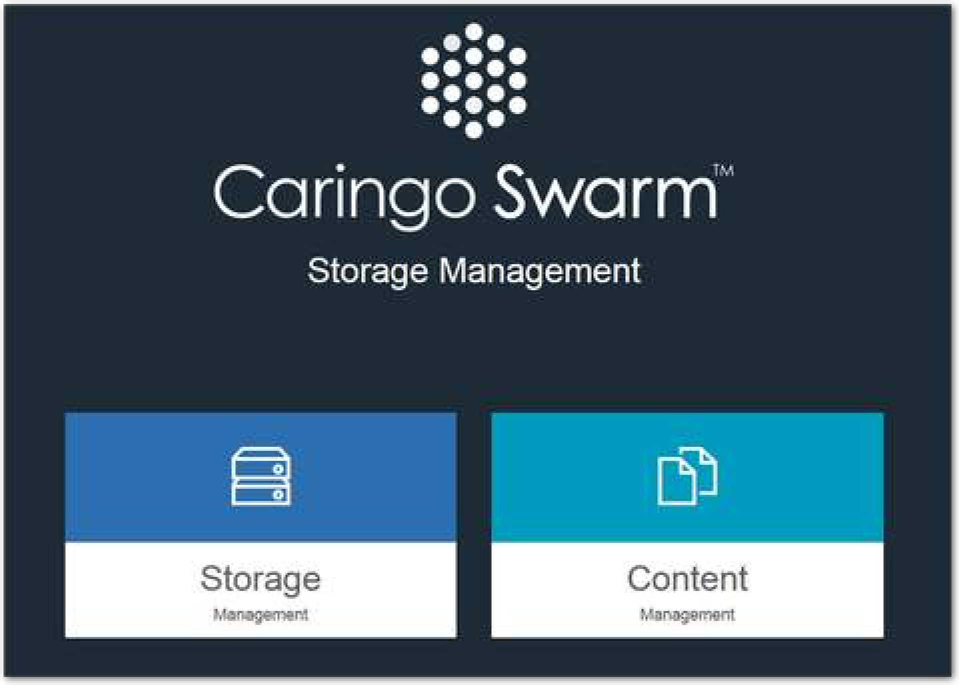
172.30.14.145

admin

••••••••••••••••••••

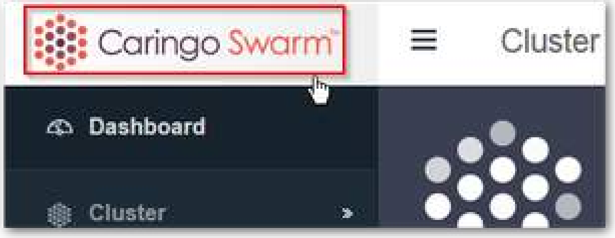
Navigating the UI

Upon logging in, you chose which Storage UI you want to use (Swarm Storage or the Content UI Overview):



Tip

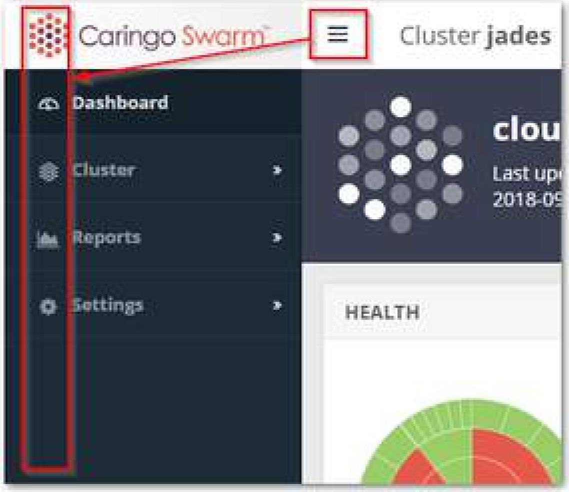
To return to this page at any time to go to the other UI, click the top of the site map (the Caringo Swarm logo):



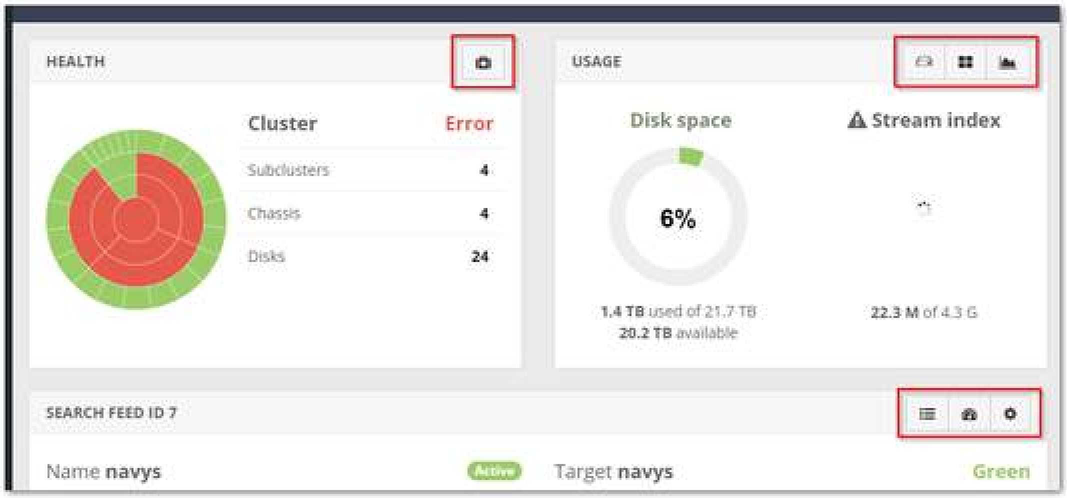
The primary navigation for the Swarm UI is in the left side navigation pane, which includes 3 major sections:

- Cluster: provides access to the details of your cluster's hardware (physical chassis), subclusters (if any), and feeds (search and replication).
- Reports: includes valuable real time and historical views into the health of and activity in your cluster.
- Settings: displays cluster-wide settings and license information.

The left pane can be opened and collapsed using the stacked bar icon next to the Caringo Swarm logo.

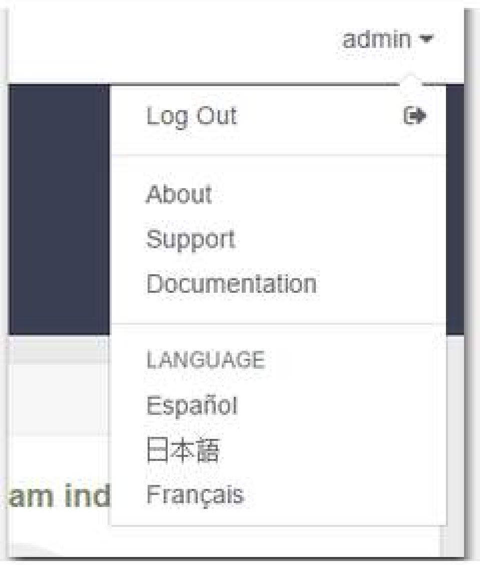


Throughout the application, quick access links to relevant reports and pages can be accessed through icons on each page. For instance, the Health section of the Dashboard includes a link to the more detailed Health Report.



Resources

Located at the top right of the Storage UI is your account name, which drops down a menu of resources:



- Log Out – ends the current session
- About – reports the version of the software in use with links to the end user license agreement and third-party licenses (requires a login to Caringo Connect)
- Support – opens the Caringo Support site (requires an account for full access)
- Documentation – opens searchable online help (requires a login to Caringo Connect)
- Language - allows selection of an alternate display language for the user interface.

> Note
> Selecting an alternate language from the resource menu will localize most but not all text. Log messages, error messages, setting descriptions, and some chart labels are not localized as they come from sources outside the UI. Setting your language preference through your browser may also return

slightly better translation results.

Rebranding the UI

To update the UI look and feel for your organization, use the 'custom' folder that comes with Swarm UI and overwrite the included files.

1. Place all of your custom files, SVGs, and stylesheets into this 'custom' directory.
2. Replace the images provided or create new images and update the stylesheet to reflect any naming changes.
3. Uncomment custom.css in order for the styling updates to work.

If the 'custom' folder does not exist, create a new folder called 'custom' and place it at the top level of the installed JavaScript files, alongside such folders as app, fonts, scripts, styles.

Viewing and Managing the Cluster

Deprecated

The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Dashboard
- Hardware
- Restarting or Shutting Down the Cluster
- Suspending or Enabling Disk Recovery
- Subclusters

The Cluster section gives you access to the details of your cluster's Hardware (physical or virtual machines, or chassis), Subclusters, and Feeds (search and replication).

Streams versus Objects

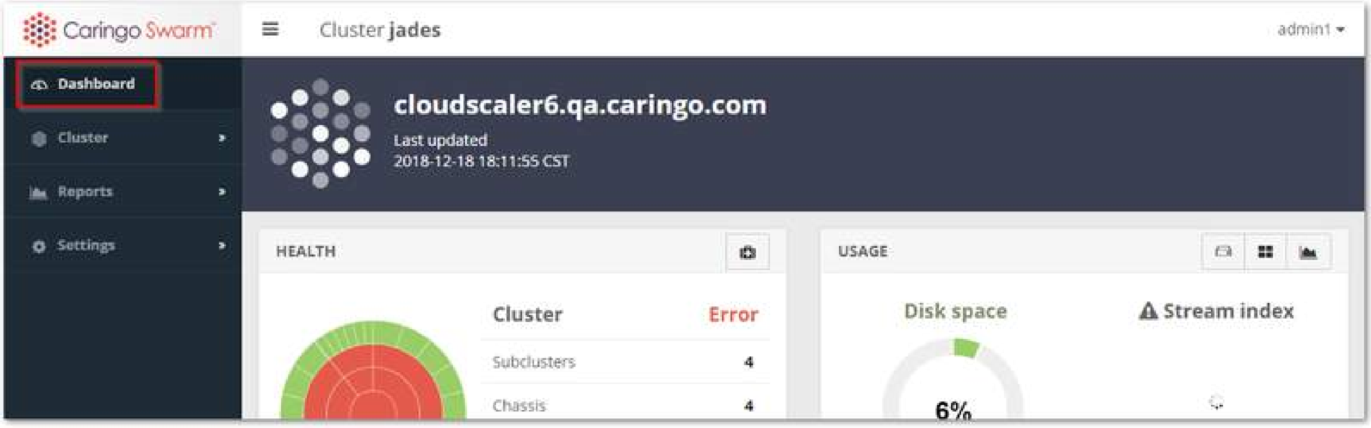
Streams are not objects. The term stream indicates a data component being managed on disk (such as one of several erasure-coded segments or replicas), not a user-facing logical object, such as appears in the Content UI by name or UUID. Many streams may comprise one object.

Dashboard

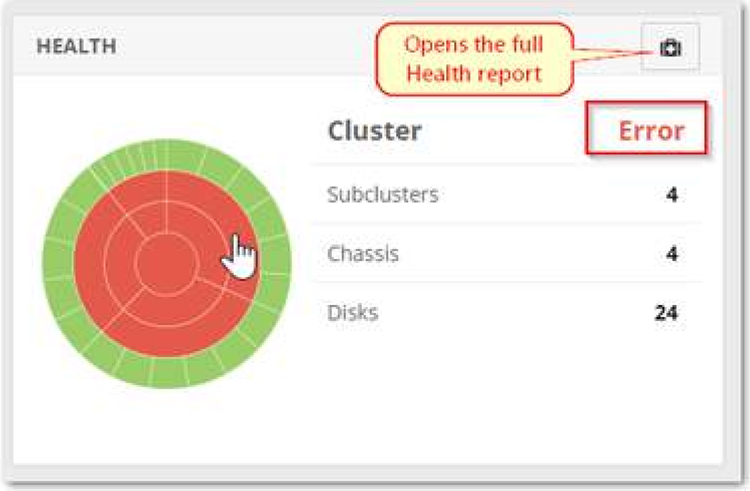
The primary initial view of the cluster is offered through the Dashboard. The Dashboard presents real-time visual monitoring, alerting, and history across the cluster's usage and its activity. The host name and last refresh time for the page are shown at the top. The page will auto-refresh every 60 seconds.

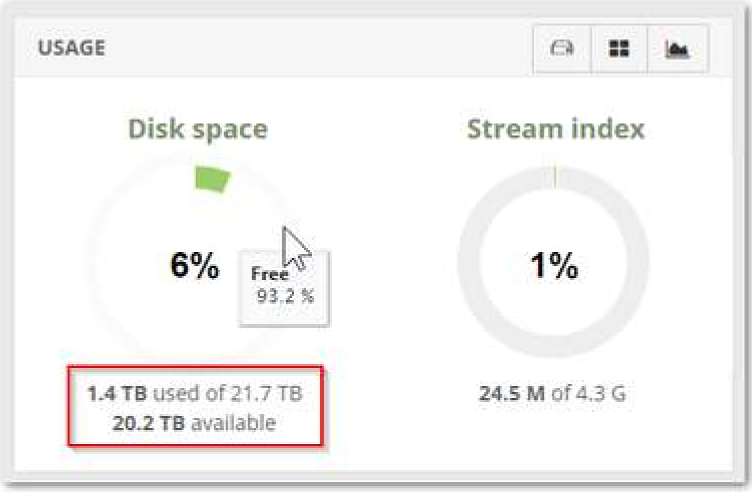
Empty dashboard

If the cluster does not have an active license, the top of the page will simply show a Caringo watermark background. If the license is active but the data is still missing, try logging out and logging in again.

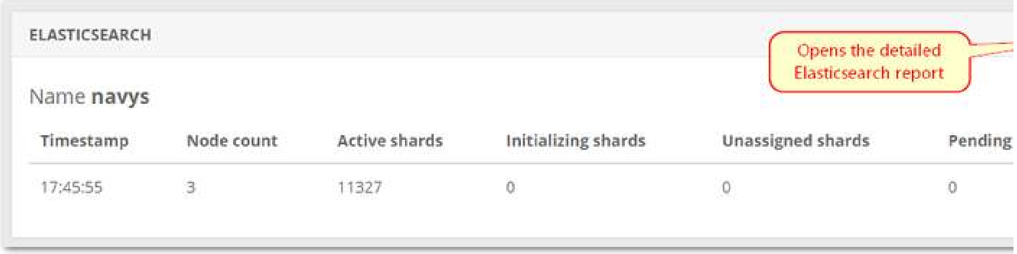


The Dashboard provides quick reference to these key indicators for the cluster:

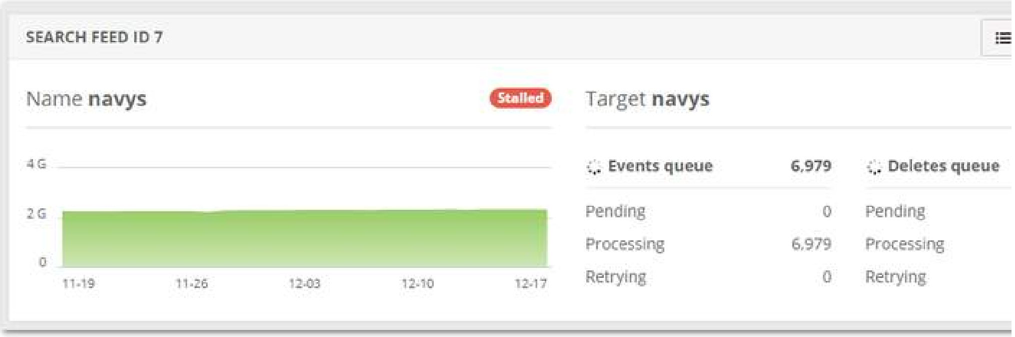
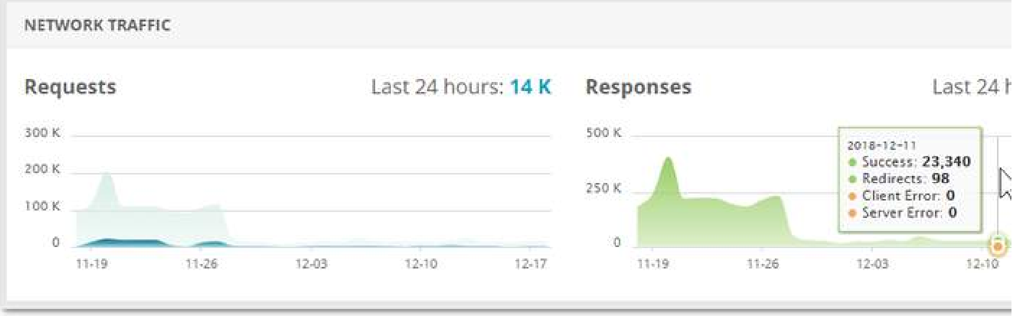
Health	Cluster	Overall cluster health as a red, yellow, green status based on the status of the cluster's c for additional details.  HEALTH Opens the full Health report Cluster Error Subclusters 4 Chassis 4 Disks 24 Tip If you cannot distinguish among those colors, hover over sections to read the valu Note Per-object feed errors (for replication or search indexing) do not generate critical l state to 'error'. Check the Feeds section of the Dashboard for blocked feeds.
Usage	Disk space	% of Disk space usage in the cluster, including free space, used space and trapped space Disk space pie chart will turn orange and then red if the capacity exceeds 90%. Mouse ov information.

<table>
<tr><td></td><td>Stream index</td><td>% of Stream index usage in the cluster, including how much is used by the overlay index i the Stream index pie chart will turn orange and then red if the capacity exceeds 90%. Mo information.


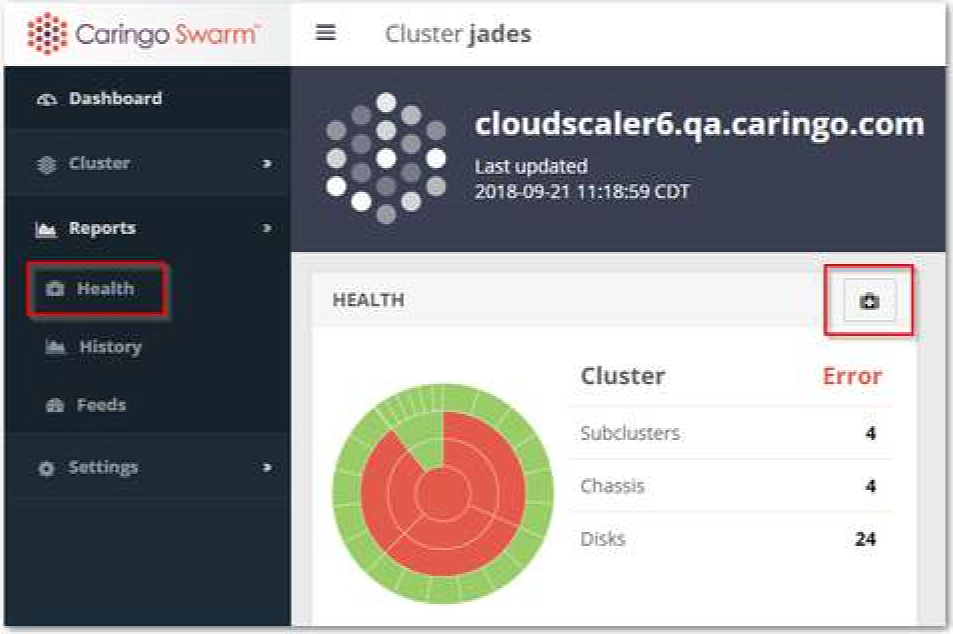
USAGE
Disk space
6%
Free
93.2 %
Stream index
1%

1.4 TB used of 21.7 TB
20.2 TB available
24.5 M of 4.3 G</td></tr>
<tr><td>Elasticsearch</td><td>Statistics</td><td>The Elasticsearch panel gives a snapshot of the ES cluster's status and essential statisti


Name navys
<table><tr><th>Timestamp</th><th>Node count</th><th>Active shards</th><th>Initializing shards</th><th>Unassigned shards</th><th>Pending</th></tr><tr><td>17:45:55</td><td>3</td><td>11327</td><td>0</td><td>0</td><td>0</td></tr></table></td></tr>
</table>

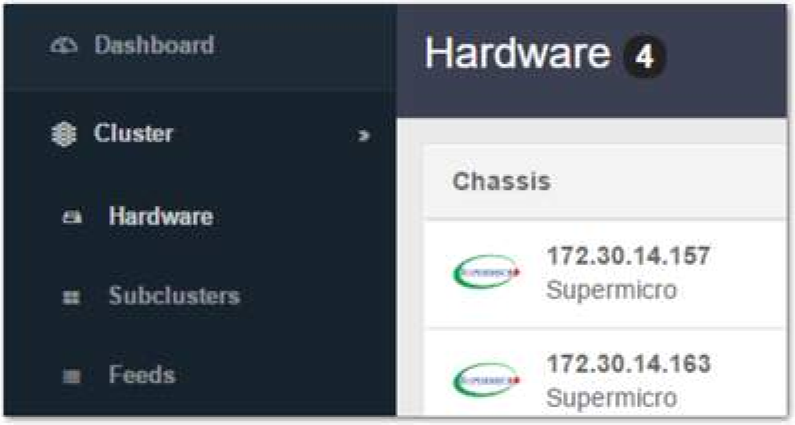
Feeds	One pane per feed	Status for each feed, both search and replication, including current feed status, a trendec count by stage (Pending, Processing, Retrying) of any queued events and deletes. If queu animated spinning icon will display to indicate that the feed is currently processing. Note It may take over 24 hours before the processing trend is visible in the graph.  SEARCH FEED ID 7 Name **navys** Stalled Target **navys** Events queue 6,979 Pending 0 Processing 6,979 Retrying 0 Deletes queue Pending Processing Retrying 4 G 2 G 0 11-19 11-26 12-03 12-10 12-17
Network Traffic	Request	The count of each SCSP method type in incoming client requests over a rolling 30 day wi
	Responses	The count of HTTP response codes returned to clients by the storage cluster over a rollir  NETWORK TRAFFIC Requests Last 24 hours: 14 K Responses Last 24 h 2018-12-11 Success: 23,340 Redirects: 98 Client Error: 0 Server Error: 0 300 K 200 K 100 K 0 11-19 11-26 12-03 12-10 12-17 500 K 250 K 0 11-19 11-26 12-03 12-10

The collapsible global menu pane provides navigation to more detailed information and reports, or you may use the icons on each section of the dashboard to drill down into details for that section. For instance, clicking the medical bag icon in the Dashboard's Health section opens the same page as selecting Reports > Health from the menu:

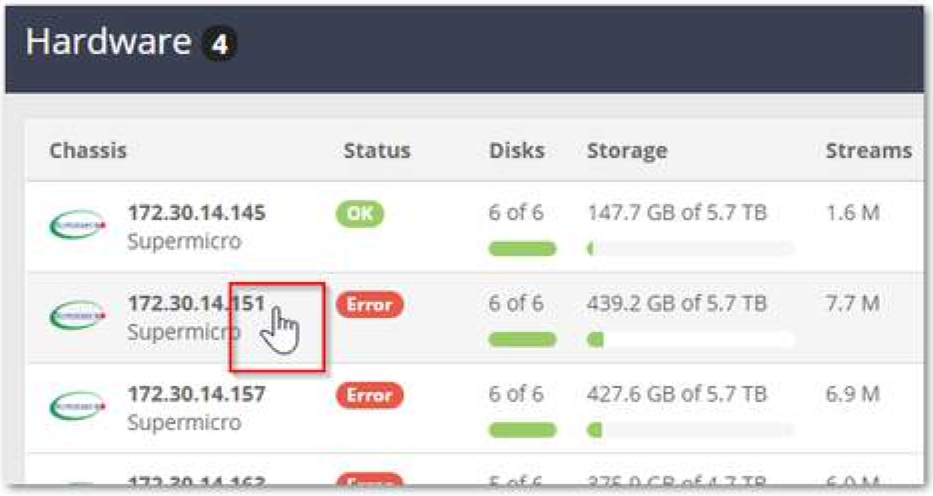


Hardware

The Hardware report includes a summary view of each server chassis in the storage cluster and its current state, including the number of chassis disks online, the used capacity, stream count, up-time, Swarm storage software version, and the subcluster to which the chassis belongs, if any.



Click on any row to drill down into the Chassis Details page for that particular chassis (physical or virtual machine):



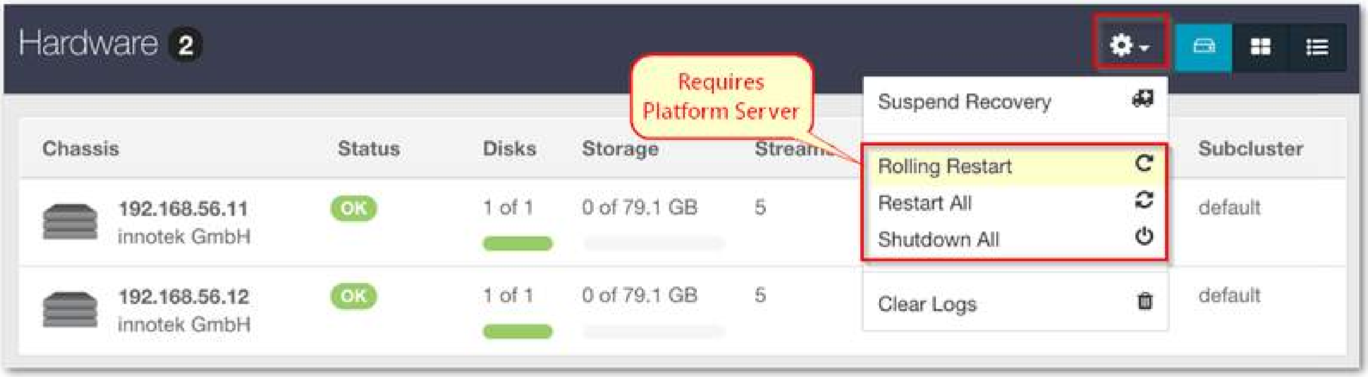
Hardware 4

Chassis	Status	Disks	Storage	Streams
172.30.14.145 Supermicro	OK	6 of 6	147.7 GB of 5.7 TB	1.6 M
172.30.14.151 Supermicro	Error	6 of 6	439.2 GB of 5.7 TB	7.7 M
172.30.14.157 Supermicro	Error	6 of 6	427.6 GB of 5.7 TB	6.9 M
172.30.14.163	Error	5 of 6	375.0 GB of 4.7 TB	6.0 M

Restarting or Shutting Down the Cluster

The settings gear icon at the top of the page allows restarting or shutting down the entire storage cluster, as well as the ability to clear logs.

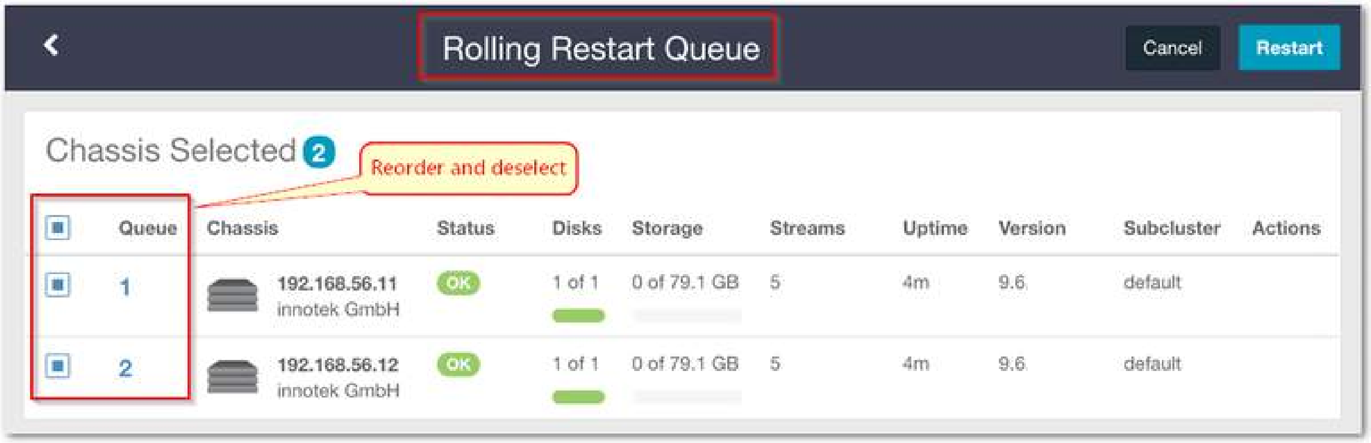
Admin only
These cluster-wide actions require authentication as an administrator.



Hardware 2

Chassis	Status	Disks	Storage	Streams	Subcluster
192.168.56.11 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	default
192.168.56.12 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	default

If you have Platform Server installed, you can perform a Rolling Restart of the cluster, with full control over the restart queue, to reorder and cancel individual chassis restarts. (v2.0)

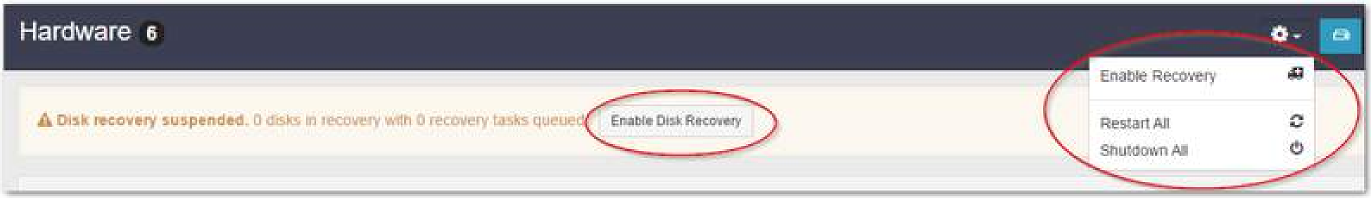


☑	Queue	Chassis	Status	Disks	Storage	Streams	Uptime	Version	Subcluster	Actions
☑	1	192.168.56.11 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	4m	9.6	default	
☑	2	192.168.56.12 innotek GmbH	OK	1 of 1	0 of 79.1 GB	5	4m	9.6	default	

Suspending or Enabling Disk Recovery

Administrators can suspend an in process disk recovery using the Suspend Recovery option under the settings gear icon.

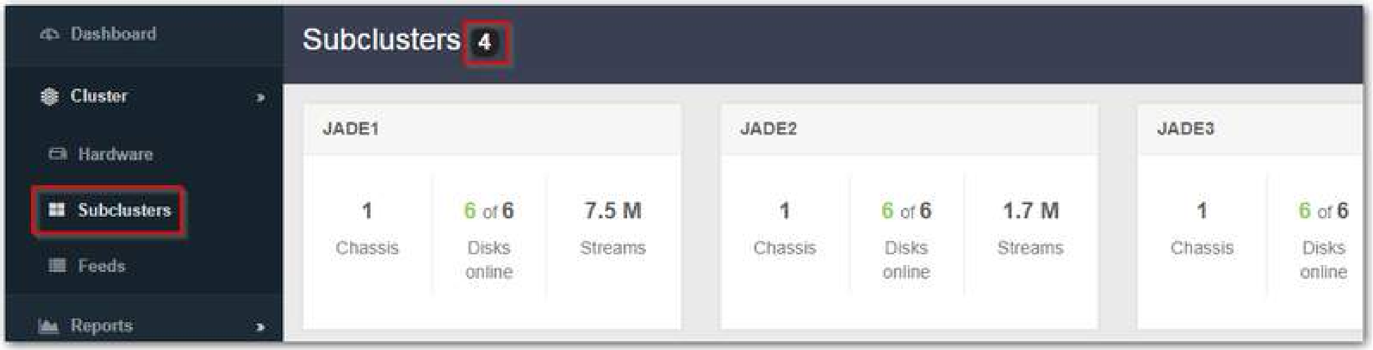
Later, you can enable a previously suspended recovery using either the Enable Disk Recovery button in the banner message or the Enable Recovery under the settings gear icon.



⚠ Disk recovery suspended. 0 disks in recovery with 0 recovery tasks queued.

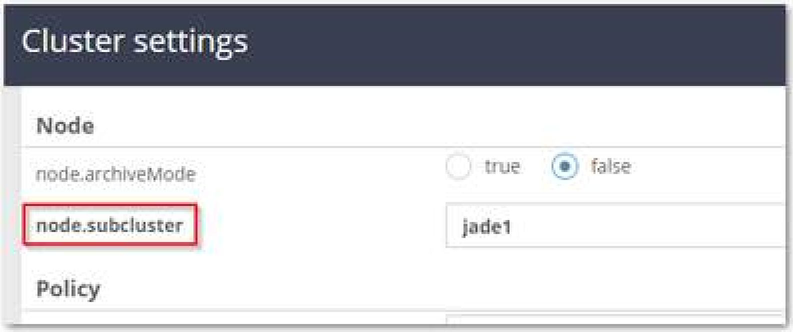
Subclusters

The Subclusters page rolls up the information on all of the subclusters that exist in your cluster, including their chassis and disk counts and number of streams.



JADE1			JADE2			JADE3	
1 Chassis	6 of 6 Disks online	7.5 M Streams	1 Chassis	6 of 6 Disks online	1.7 M Streams	1 Chassis	6 of 6 Disks online

To dynamically change a subcluster assignment, go to Hardware > Chassis Details and click the Settings tab. (Subclusters can be configured from the CSN (cluster.cfg) or in the node's configuration file (node.cfg), but these require a cluster reboot to take effect.)



Cluster settings

Node

node.archiveMode ○ true ● false

node.subcluster jade1

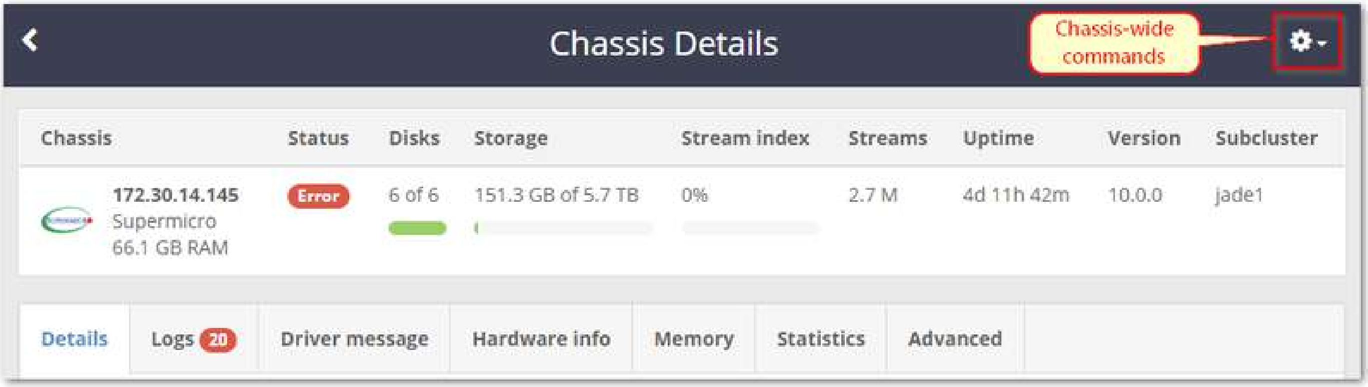
Policy

Managing Chassis and Drives

Deprecated
The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

Chassis Details

Detailed hardware and status information for each chassis (physical or virtual machine) are displayed on the hardware details page.



Chassis Details

Chassis-wide commands

Chassis	Status	Disks	Storage	Stream index	Streams	Uptime	Version	Subcluster
172.30.14.145 Supermicro 66.1 GB RAM	Error	6 of 6	151.3 GB of 5.7 TB	0%	2.7 M	4d 11h 42m	10.0.0	jade1

Streams, not objects

Keep in mind that streams are counts of the total number of Swarm-managed data components (such as replicas and segments). Streams are not logical objects (such as video files).

Status states — These are the states you might see reported for hardware in your cluster and how to interpret them:

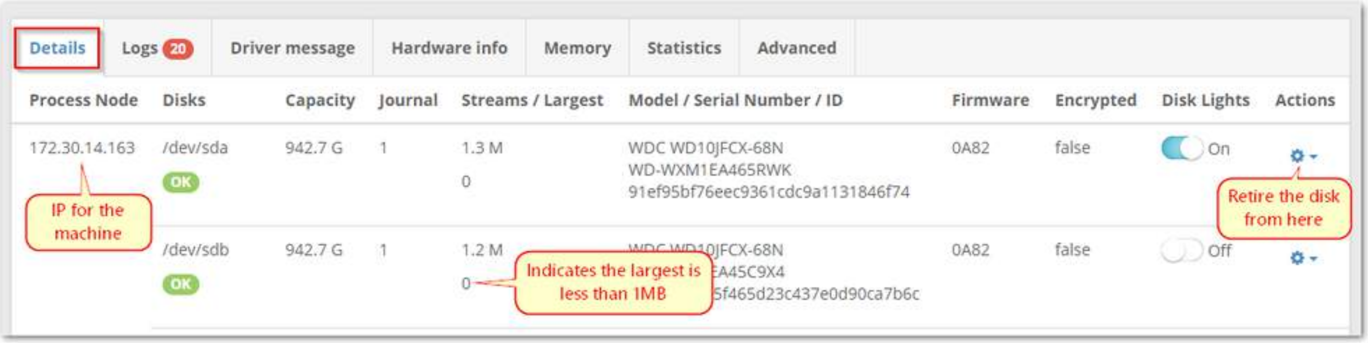
Status	Nodes / Chassis	Volumes / Disks
ok	Nominal	Nominal
idle	Nominal, but the node is idle	Nominal, but idle
retiring	One or more volumes are offloading its streams to the cluster due to retire	Offloading its streams to the cluster due to retire
retired	All volumes are retired	Empty of objects and not taking new ones
unavailable		In an error state
error	Errors have been reported on the node (hardware or software)	
mounting	One or more volumes are mounting	Mounting at startup/discovery
finalizing	Can appear while the node is rebooting or shutting down, as the node finishes sessions in process	
maintenance	A 3-hour window during an administrative reboot or shutdown where Failed Volume Recovery will not run	

Details Tab

Each detailed row displays a drive name, status, total capacity, amount of used journal space, the largest stream size it contains in MB, Model number, Serial Number, ID, Firmware version, and Encryption status. If the largest stream on disk is less than 1MB, the Largest value will display as 0.

Tip

When you are retiring a disk, watch the Streams count to track the progress.



Process Node	Disks	Capacity	Journal	Streams / Largest	Model / Serial Number / ID	Firmware	Encrypted	Disk Lights	Actions
172.30.14.163	/dev/sda OK	942.7 G	1	1.3 M 0	WDC WD10JFCX-68N WD-WXM1EA465RWK 91ef95bf76eec9361cdc9a1131846f74	0A82	false	On	⚙
	/dev/sdb OK	942.7 G	1	1.2 M 0	WDC WD10JFCX-68N EA45C9X4 5f465d23c437e0d90ca7b6c	0A82	false	Off	⚙

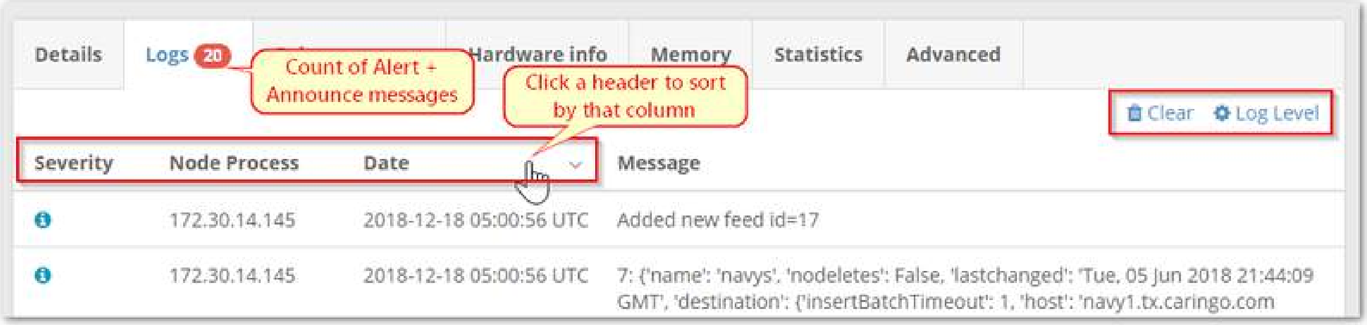
IP for the machine

Indicates the largest is less than 1MB

Retire the disk from here

Logs Tab

The Logs tab lists the last 10 logged announcements in the cluster as well as the last 10 logged critical alerts. The tab itself includes a count of these messages, and it will appear red if any of them are errors:

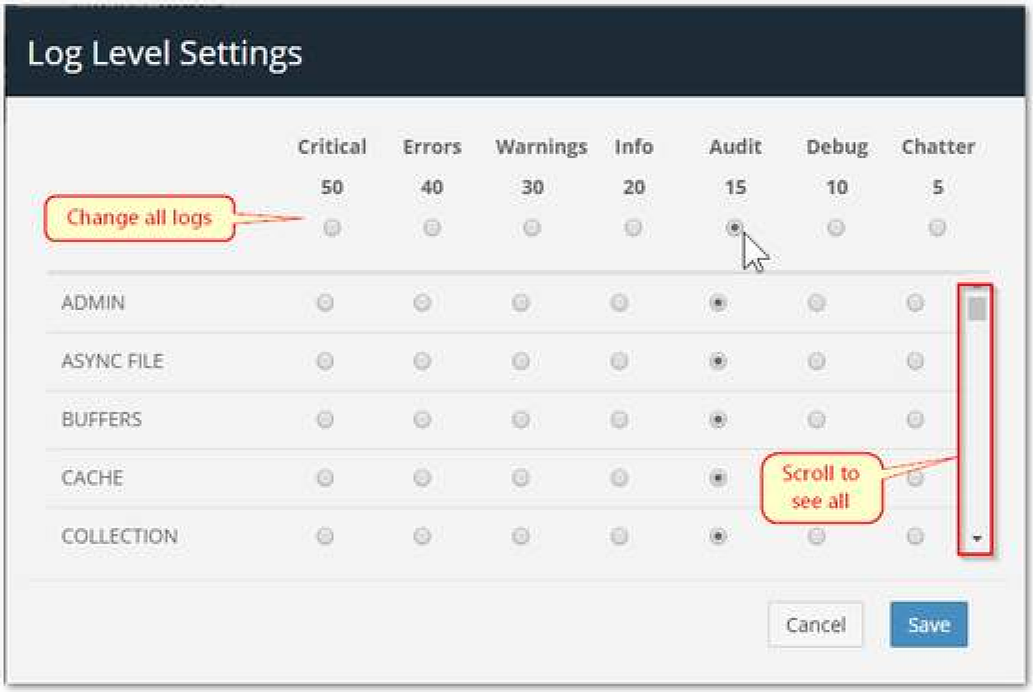


Count of Alert + Announce messages

Click a header to sort by that column

Severity	Node Process	Date	Message
ℹ	172.30.14.145	2018-12-18 05:00:56 UTC	Added new feed id=17
ℹ	172.30.14.145	2018-12-18 05:00:56 UTC	7: {'name': 'navys', 'nodeletes': False, 'lastchanged': 'Tue, 05 Jun 2018 21:44:09 GMT', 'destination': {'insertBatchTimeout': 1, 'host': 'navy1.tx.caringo.com

- To remove log messages that have either been addressed or are not interesting from the display, use the Clear command.
- To view and change the log levels set for this machine, click the Log Level (gear) settings command.



Log Level Settings

	Critical 50	Errors 40	Warnings 30	Info 20	Audit 15	Debug 10	Chatter 5
Change all logs	○	○	○	○	●	○	○
ADMIN	○	○	○	○	●	○	○
ASYNC FILE	○	○	○	○	●	○	○
BUFFERS	○	○	○	○	●	○	○
CACHE	○	○	○	○	●		○
COLLECTION	○	○	○	○	●	○	○

Scroll to see all

Hot-swapping drives — Messages display on this tab if you remove or insert a drive into a running node. This feature, referred to as Hot Swapping and Plugging Drives, lets you remove failed drives for analysis or to add storage capacity to a node at any time.

For example, if you add a volume and then remove that volume, the following messages appear:

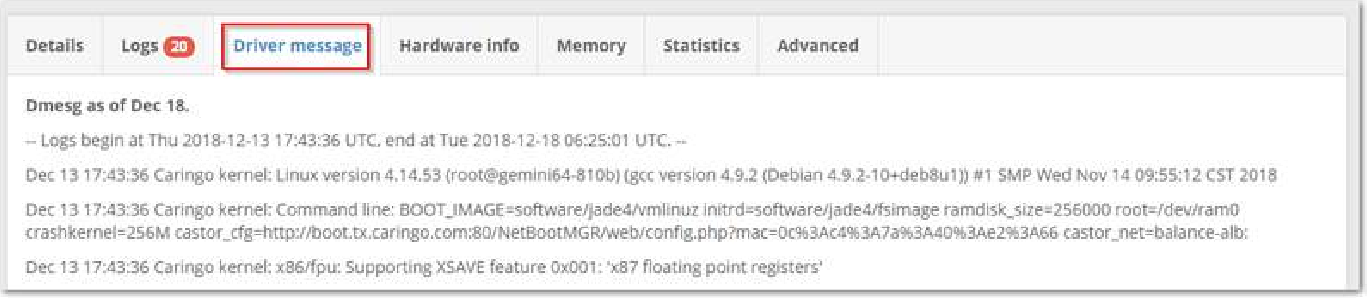
```
mounted /dev/sdb, volumeID is 561479FB832DCC526B1D7EDCD06B83E1
removed /dev/sdb, volumeID was 561479FB832DCC526B1D7EDCD06B83E1
```

Note
These messages appear at the announcement level. Additional debug level messages appear in the syslog.

Driver Message Tab

dmesg (driver message) is a Linux command that prints the message buffer of the kernel. These driver messages are useful for diagnosing a Swarm issue when a system panic or error occurs.

Note
dmesg is a circular buffer, so it shows only the last 1000 kernel messages.

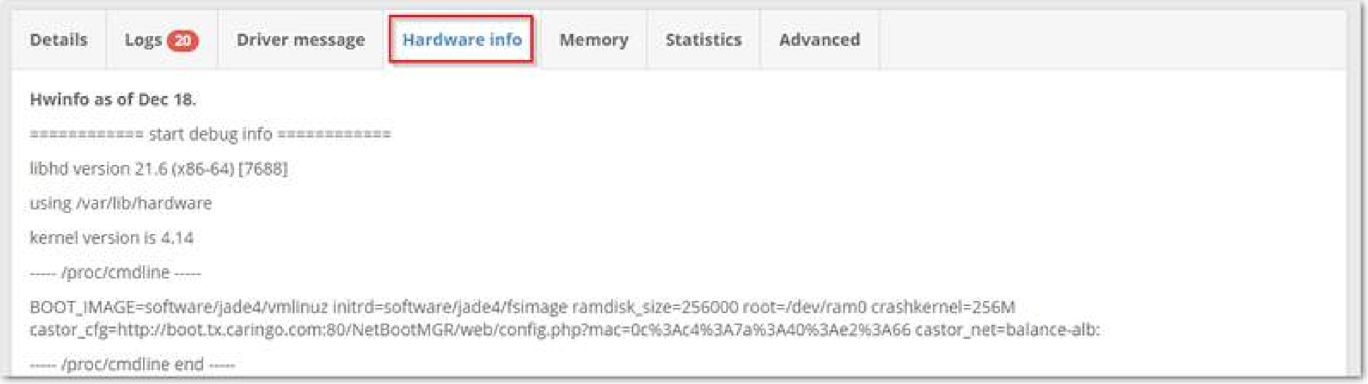


Dmesg as of Dec 18.

```
-- Logs begin at Thu 2018-12-13 17:43:36 UTC, end at Tue 2018-12-18 06:25:01 UTC. --

Dec 13 17:43:36 Caringo kernel: Linux version 4.14.53 (root@gemini64-810b) (gcc version 4.9.2 (Debian 4.9.2-10+deb8u1)) #1 SMP Wed Nov 14 09:55:12 CST 2018

Dec 13 17:43:36 Caringo kernel: Command line: BOOT_IMAGE=software/jade4/vmlinuz initrd=software/jade4/fsimage ramdisk_size=256000 root=/dev/ram0
crashkernel=256M castor_cfg=http://boot.tx.caringo.com:80/NetBootMGR/web/config.php?mac=0c%3Ac4%3A7a%3A40%3Ae2%3A66 castor_net=balance-alb:

Dec 13 17:43:36 Caringo kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
```

Hardware Info Tab

hwinfo (hardware information) is the Linux hardware detection tool output. This tool probes for the hardware present in the system and displays detailed information about various hardware components in human-readable format.



Hwinfo as of Dec 18.

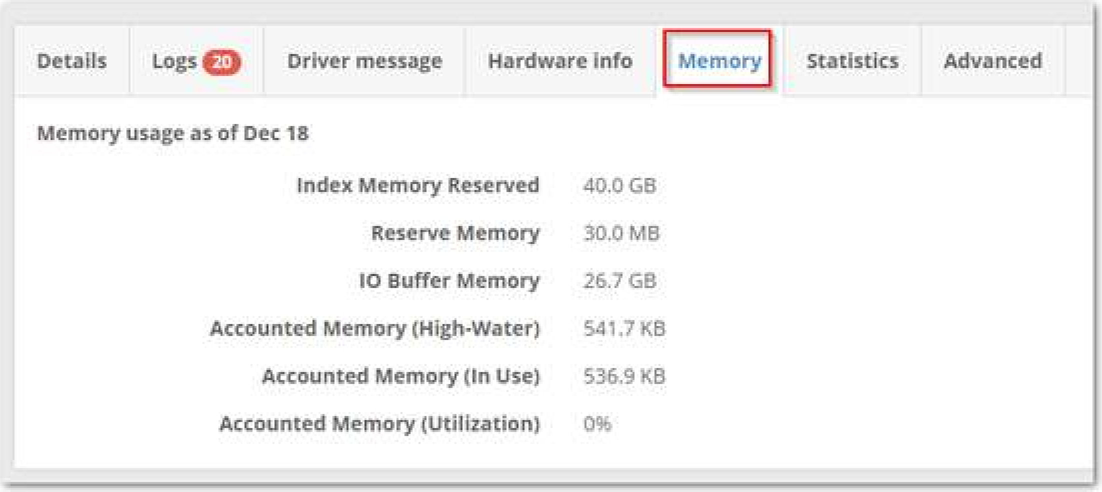
```
============ start debug info ============

libhd version 21.6 (x86-64) [7688]

using /var/lib/hardware

kernel version is 4.14

----- /proc/cmdline -----

BOOT_IMAGE=software/jade4/vmlinuz initrd=software/jade4/fsimage ramdisk_size=256000 root=/dev/ram0 crashkernel=256M
castor_cfg=http://boot.tx.caringo.com:80/NetBootMGR/web/config.php?mac=0c%3Ac4%3A7a%3A40%3Ae2%3A66 castor_net=balance-alb:

----- /proc/cmdline end -----
```

Memory Tab

The usage report on the Memory tab provides detailed information to help with troubleshooting insufficient memory. Each node uses memory in order to hold an index of the objects stored in it. If a node runs out of index space, it will stop storing new content until space is freed through deletions. However, a full node continues to respond to client read requests for data that is already on it. Each named or alias object requires two index slots. Erasure coding typically requires more memory than replication; exactly how much depends on the encoding.

Best practice

If you are running out of index slots through normal activity, increase the memory in the node.

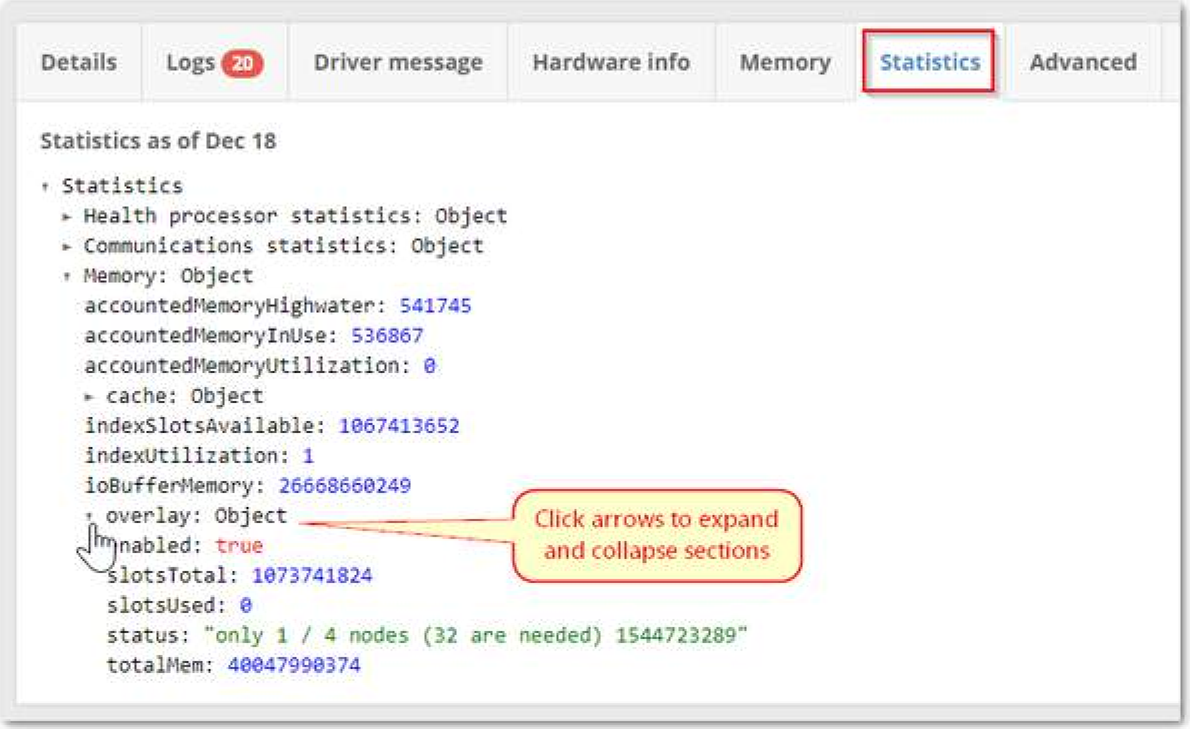


Memory usage as of Dec 18

Index Memory Reserved	40.0 GB
Reserve Memory	30.0 MB
IO Buffer Memory	26.7 GB
Accounted Memory (High-Water)	541.7 KB
Accounted Memory (In Use)	536.9 KB
Accounted Memory (Utilization)	0%

Statistics Tab

The Statistics tab rolls up a detailed, expandable report that combines Health Processor (HP), Communications (cluster network), and Memory usage counts and values, to help with analysis and troubleshooting.



Statistics as of Dec 18

```
▾ Statistics
  ▸ Health processor statistics: Object
  ▸ Communications statistics: Object
  ▾ Memory: Object
    accountedMemoryHighwater: 541745
    accountedMemoryInUse: 536867
    accountedMemoryUtilization: 0
    ▸ cache: Object
    indexSlotsAvailable: 1067413652
    indexUtilization: 1
    ioBufferMemory: 26668660249
    ▾ overlay: Object
      enabled: true
      slotsTotal: 1073741824
      slotsUsed: 0
      status: "only 1 / 4 nodes (32 are needed) 1544723289"
      totalMem: 40047990374
```

Click arrows to expand and collapse sections

The health processor runs on each Swarm node to check the status of streams, performing a wide range of actions:

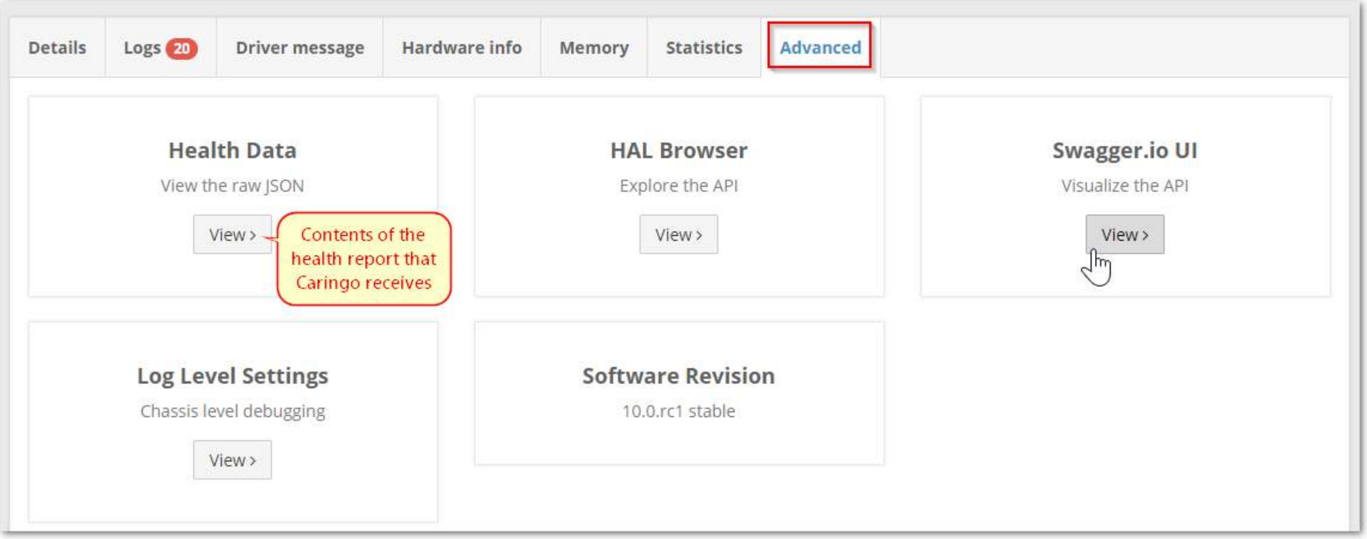
- Sends replica checks to the other nodes and based on that adds or trims replicas
- Deletes streams that require deletion according to lifepoints
- Provides a safety net to remove older alias and named stream versions when a newer version is found in the cluster (which can happen when nodes are restored)
- Checks each stream for data corruption via comparison with the stored stream hash

- Moves the stream on disk, if defragmentation is needed
- Ensures that the disk index is consistent with the streams found on disk
- Ensures that replicas are distributed properly in the cluster

Advanced Tab

The Advanced tab lets you dynamically change machine-level logging levels and also work with Swarm's management API, both through a hands-on HAL browser and a Swagger visualizer.

The Health Data is the raw JSON content of the health report that your cluster sends to Caringo Support. See Health Data to Support.



Health Data
View the raw JSON

Contents of the health report that Caringo receives

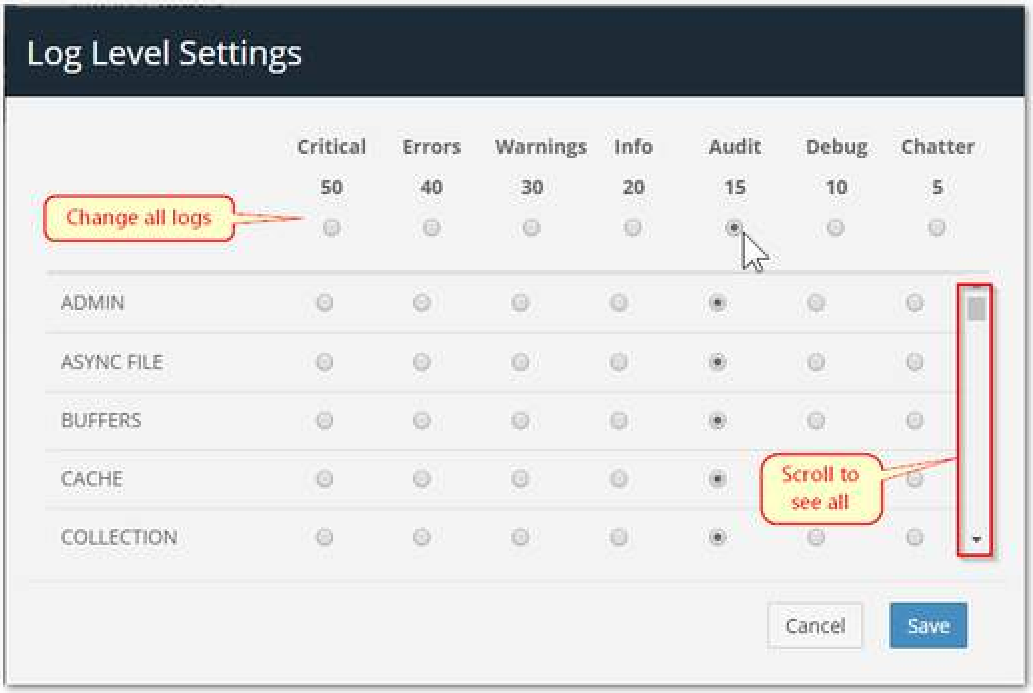
HAL Browser
Explore the API

Swagger.io UI
Visualize the API

Log Level Settings
Chassis level debugging

Software Revision
10.0.rc1 stable

You can reset the log levels from this tab, as well as from the Logs tab:



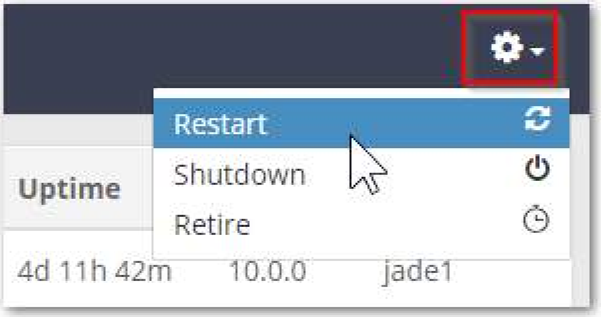
Log Level Settings

	Critical 50	Errors 40	Warnings 30	Info 20	Audit 15	Debug 10	Chatter 5
Change all logs	○	○	○	○	●	○	○
ADMIN	○	○	○	○	●	○	○
ASYNC FILE	○	○	○	○	●	○	○
BUFFERS	○	○	○	○	●	○	○
CACHE	○	○	○	○	●	○	○
COLLECTION	○	○	○	○	●	○	○

Scroll to see all

Restarting or Shutting Down a Chassis

The gear icon at the top of the page allows you to restart or shut down the chassis. A node that is shut down or rebooted by an Administrator will appear with a **Maintenance** state on other nodes in the cluster.

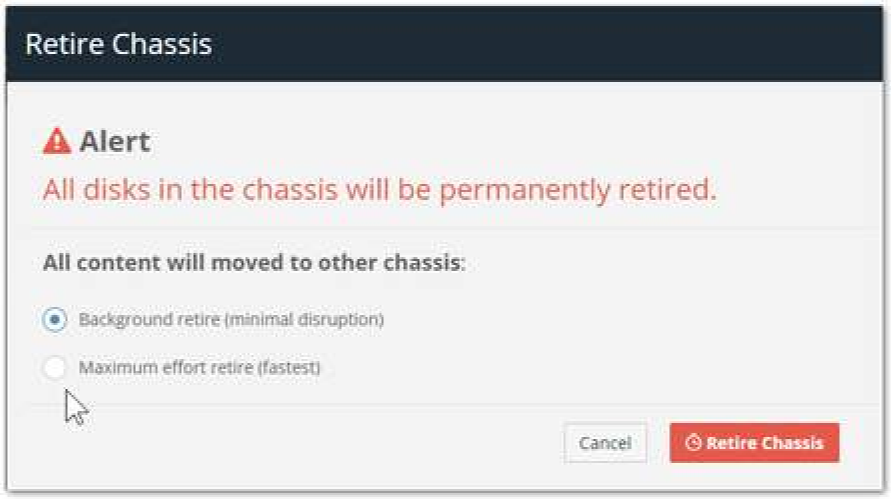


Retiring a Chassis

Whenever it's time to replace Swarm storage volumes for regular maintenance or to upgrade the cluster chassis with higher capacity drives, you need to retire the chassis. Retiring a chassis means all of its objects are copied to other chassis in the cluster, allowing you to safely remove the chassis drives without risking any data loss.

> Important
> Before you retire a chassis, make sure that the cluster:
>
> - Has enough capacity for the objects on the retiring chassis to replicate elsewhere.
> - Has enough remaining nodes to replicate the objects with only one replica on any given node.

To initiate a retire, select the Retire option under the gear icon at the top of the Chassis Details page. When you initiate a retire, you choose if you would like a minimally disruptive retire that is limited to just the chassis being retired, or an accelerated retire that uses all nodes in the cluster to replicate objects on the retiring chassis as quickly as possible. Note that the cluster-wide retire may impact performance as it does put additional load on the cluster.


Retire Chassis

Alert

All disks in the chassis will be permanently retired.

All content will moved to other chassis:

● Background retire (minimal disruption)

○ Maximum effort retire (fastest)

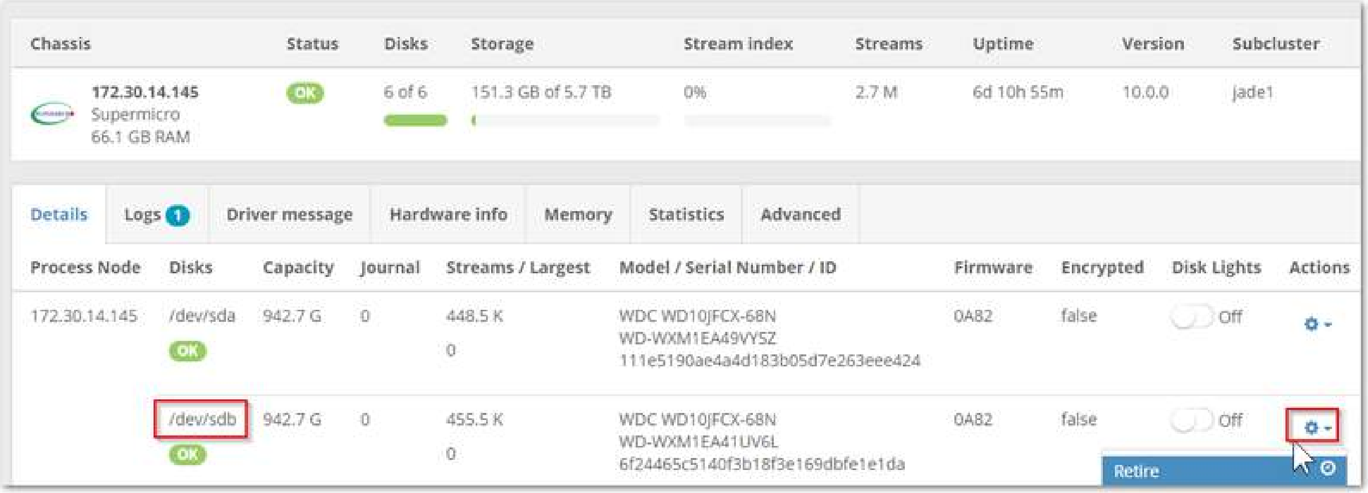
> Replica protection

> Retire succeeds only if objects can be replicated elsewhere in the cluster. As a result, the Retire action will not remove an object until it can guarantee that at least two replicas exist in the cluster or the existing number of replicas matches the policy.replicas min parameter value.

A retiring chassis accepts no new or updated objects. You can cancel an in process retire by selecting the Cancel Retire option under the gear icon at the top of the Chassis Details page. You can only cancel a retire while one or more drives in the chassis have a Retiring status. After all objects are copied elsewhere, the each chassis volume's state changes to Retired and Swarm storage no longer uses the volume. At this point, you should remove and repair the volume or discard it.

Retiring a Disk (Volume)

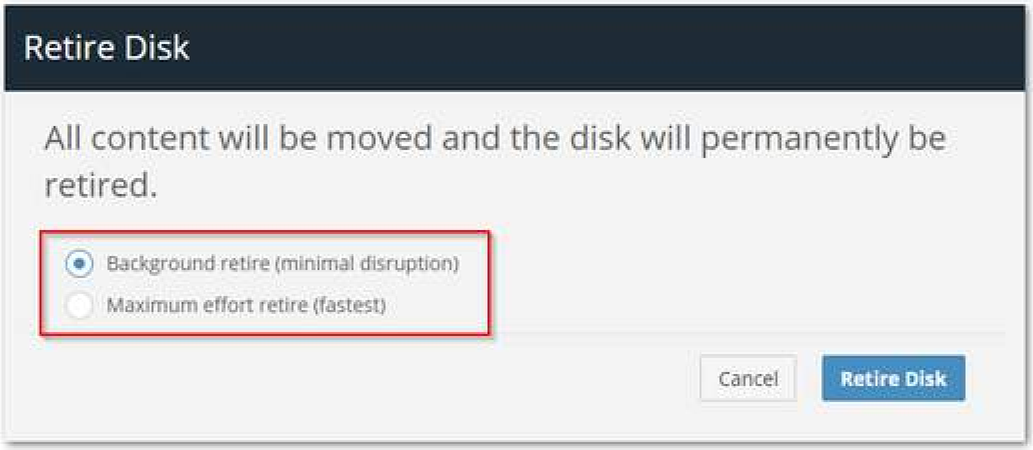
Disk-level retires are useful for targeting bad (slow) disks and for working around having too limited capacity for retires of entire chassis. To retire a volume, locate and click the gear icon in the row for the affected disk:



Chassis	Status	Disks	Storage	Stream index	Streams	Uptime	Version	Subcluster
172.30.14.145 Supermicro 66.1 GB RAM	OK	6 of 6	151.3 GB of 5.7 TB	0%	2.7 M	6d 10h 55m	10.0.0	jade1

Process Node	Disks	Capacity	Journal	Streams / Largest	Model / Serial Number / ID	Firmware	Encrypted	Disk Lights	Actions
172.30.14.145	/dev/sda OK	942.7 G	0	448.5 K 0	WDC WD10JFCX-68N WD-WXM1EA49VYSZ 111e5190ae4a4d183b05d7e263eee424	0A82	false	Off	
	/dev/sdb OK	942.7 G	0	455.5 K 0	WDC WD10JFCX-68N WD-WXM1EA41UV6L 6f24465c5140f3b18f3e169dbfe1e1da	0A82	false	Off	

Choose your speed of retire, keeping in mind that the fastest method incurs maximum effort by the cluster to move the content:

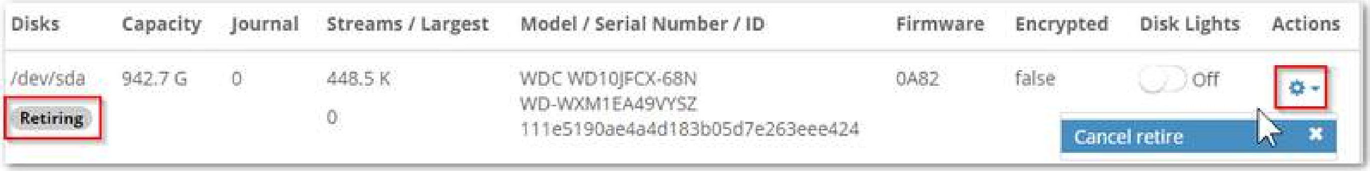


Retire Disk

All content will be moved and the disk will permanently be retired.

● Background retire (minimal disruption)
○ Maximum effort retire (fastest)

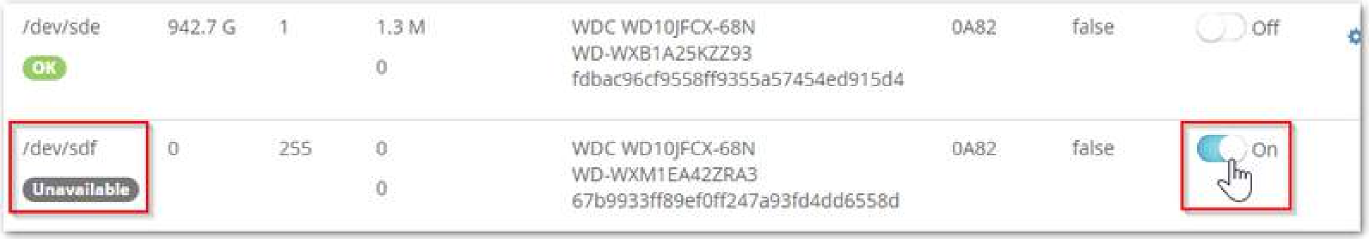
Should you need to cancel, click the gear icon in the row for the affected disk again, where you can select the Cancel retire command:



Disks	Capacity	Journal	Streams / Largest	Model / Serial Number / ID	Firmware	Encrypted	Disk Lights	Actions
/dev/sda Retiring	942.7 G	0	448.5 K 0	WDC WD10JFCX-68N WD-WXM1EA49VYSZ 111e5190ae4a4d183b05d7e263eee424	0A82	false	Off	Cancel retire

Identifying a Drive

When attempting to identify a failed or failing drive, it is helpful to enable the LED drive light for the drive. To flash the drive light for a specific drive, click on the drive light toggle in the drive's display row:



/dev/sde OK	942.7 G	1	1.3 M 0	WDC WD10JFCX-68N WD-WXB1A25KZZ93 fdbac96cf9558ff9355a57454ed915d4	0A82	false	Off
/dev/sdf Unavailable	0	255	0 0	WDC WD10JFCX-68N WD-WXM1EA42ZRA3 67b9933ff89ef0ff247a93fd4dd6558d	0A82	false	On

> Remember
>
> Drive lights will remain lit until manually turned off, so return to the Chassis Details page and click the drive light switch to Off.

Viewing and Editing Feeds

> Deprecated
>
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Adding a Search Feed
- Adding a Replication Feed
- Managing Feeds

> Note
>
> Domain and bucket context objects are prioritized for both replication and metadata feeds; this improves usability when you initiate remote sites and searching functionality.

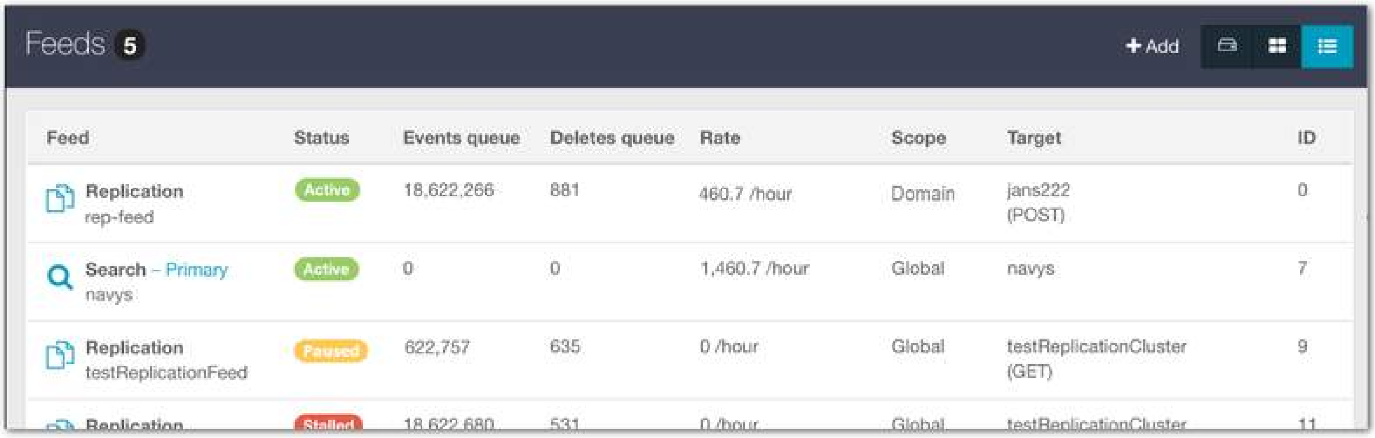
> Important
>
> A feed can appear to be idle with items still queued for processing. Plan for the fact that feed status reporting is a best-effort snapshot, not a low-latency or guaranteed transaction mechanism.

The Data Feeds report provides a summary view of the configured feeds for the storage cluster and includes their status, number of queued events and deletes, their processing rate, scope (global or domain-level), and internal feed

ID. Feed status can be:

- Active. Operating normally.
- Recovering. Temporarily paused due to volume recovery.
- Paused. Paused by user request.
- Blocked. Feed is stalled. Processing is blocked due to a transient condition.
- Configuration error. Feed is unable to operate due to incorrect configuration.
- Overlapping feeds. More than the limit of 8 feeds have been defined for the same set of objects.
- Closed. Feed is inactive and no longer operational.

Click on any feed row to view or edit the configuration for that feed.

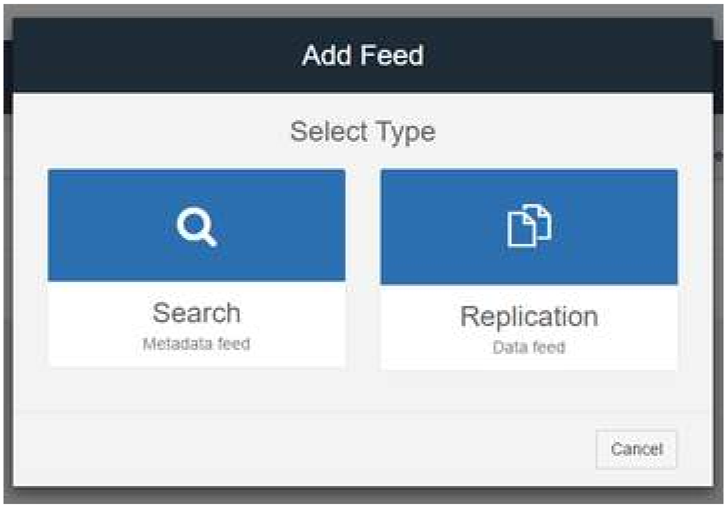


Feed	Status	Events queue	Deletes queue	Rate	Scope	Target	ID
Replication rep-feed	Active	18,622,266	881	460.7 /hour	Domain	jans222 (POST)	0
Search – Primary navys	Active	0	0	1,460.7 /hour	Global	navys	7
Replication testReplicationFeed	Paused	622,757	635	0 /hour	Global	testReplicationCluster (GET)	9
Replication	Stalled	18,622,680	531	0 /hour	Global	testReplicationCluster	11

SwarmNFS

Once a feed is defined, its settings window displays a read-only Alias field, which is the name by which Elasticsearch references the Swarm feed. Use this value as the Search Index when defining NFS exports. (v1.2.4)

To add a feed in the cluster, click the +Add button at the top of the Data Feed page and then select whether you would like to add a Search or Replication feed.



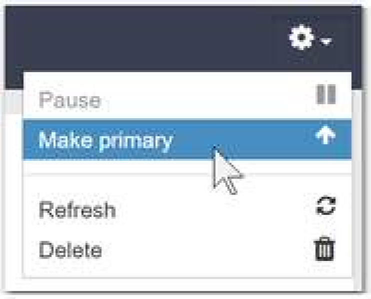
Adding a Search Feed

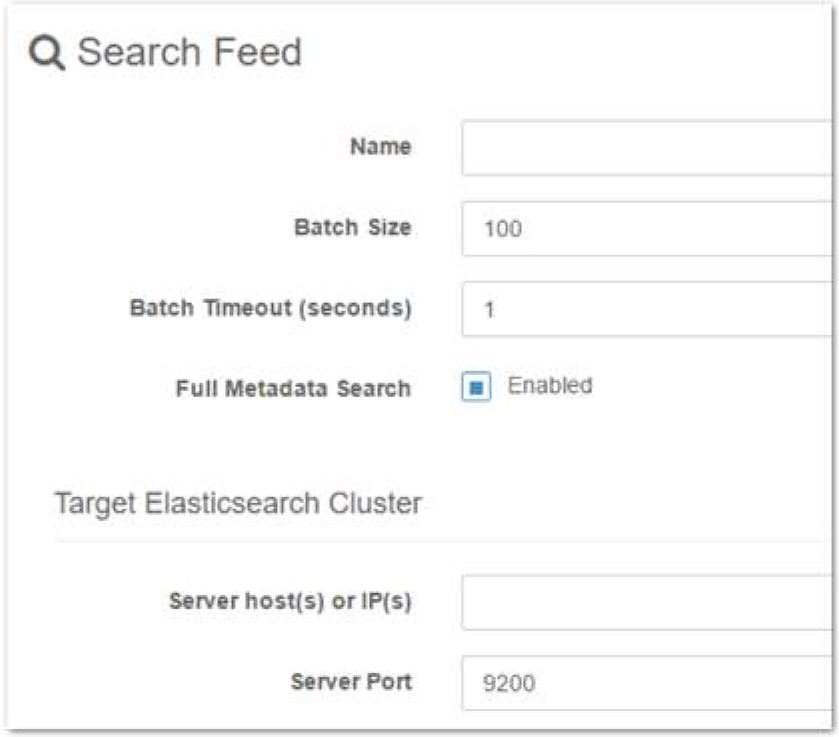
Swarm applies a naming scheme that ensures that Elasticsearch index names are always unique, within and across clusters. If you create a second search feed through the UI, Swarm creates a new Elasticsearch index and alias for that feed. (v9.0)

Multiple feeds - Swarm Storage lets you create more than one Search feed so that you can transition from using one Elasticsearch cluster to another. During the transition, continue using the primary feed for queries; the second feed is incomplete until it fully clears its backlog. When the second feed is caught up, transition to it (apply Make primary to the second feed) as soon as reasonable for your operations. When you verify that it is working as the new primary feed target, delete the original feed. Having multiple feeds is usually for temporary use only because every feed incurs cluster activity, even when paused.

Important

If you change the default search feed (apply Make primary to a new feed), you must restart all Gateway servers to pick up the new feed and update any SwarmNFS export configurations.





Search Feed

Name

Batch Size 100

Batch Timeout (seconds) 1

Full Metadata Search ■ Enabled

Target Elasticsearch Cluster

Server host(s) or IP(s)

Server Port 9200

The following table describes the data entry fields in the dialog box.

ID (For existing feeds)		Read-only; system-assigned identifier
Status (For existing feeds)	Active Primary	Read-only; the current feed processing state. The state can be: • Active. Operating normally. • Recovering. Temporarily paused due to volume recovery. • Paused. Paused by user request. • Blocked. Processing blocked due to a transient condition. • Configuration error. Feed is unable to operate due to incorrect configuration. • Overlapping feeds. More than the limit of 8 feeds have been defined for the same set of objects. • Closed. Feed is inactive and no longer operational. Flags the Search feed that will be used for all search queries. Only one feed can be Primary. Set from the Feeds command menu.
Name		The name you attach to this feed.
Batch Size	100	The maximum number of objects that would be sent concurrently to be processed.
Batch Timeout (seconds)	1	The maximum amount of time (in seconds) before a batch is resent to be processed after a timeout.

Search Full Metadata	Enabled	Enabled - Swarm storage indexes all object metadata, including baseline and custom metadata fields. Disabled - Swarm storage indexes only the baseline metadata fields. See Metadata Field Matching for a list of baseline and custom fields.
Server Host(s) or IP(s)		The IP addresses or server names that are resolvable by DNS. If you enter more than one, separate them with a comma or space. DNS must be configured on both the source and target clusters. Important If you change the list of ES servers on an active feed, be sure to refresh the feed to prevent it becoming blocked.
Server Port	9200	The default port for a host.

Adding a feed to an existing index

If you need to create a new search feed that points to an existing index, use one of these methods:

Create alias

1. Create a new ES alias that points to the existing index:

   ```
   curl -i -X POST <es- node>:9200/_aliases \
   -d '{
      "actions": [
         {
            "add": {
               "index": "<existing- index- name>",
               "alias": "<new- alias- name>"
            }
         }
      ]
   }'
   ```

2. Create a new index feed and specify the new ES alias just created:

   ```
   curl -i -X POST --anyauth -u admin:ourpwdofchoicehere
   <swarm- node>:91/api/storage/feeds \
   -d '{
      "actions": [
         {
            "add": {
               "index": "<existing- index- name>",
               "alias": "<new- alias- name>"
            }
         }
      ]
   }'
   ```

Remap alias

1. Create an index feed through the UI as usual.
2. Remap the ES alias to the existing index:

```
curl -i -X POST <es- node>:9200/_aliases \
-d '{
   "actions": [
      {
         "remove": {
            "index": "<new- index- name>",
            "alias": "<new- alias- name>"
         }
      },
      {
         "add": {
            "index": "<existing- index- name>",
            "alias": "<new- alias- name>"
         }
      }
   ]
}'
```

3. Delete the new index created for the new feed:

```
curl -i -X DELETE <es- node>:9200/<new- index- name>
```

Adding a Replication Feed

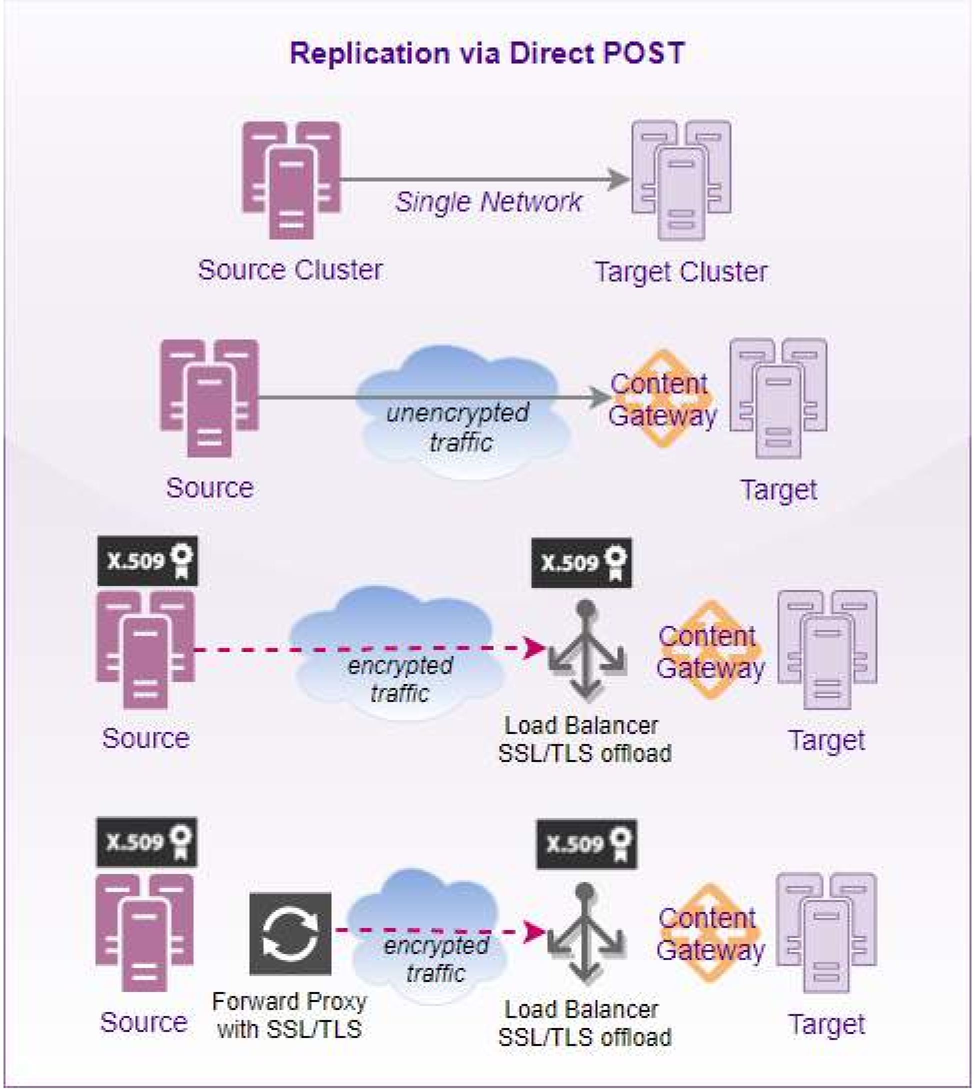
What type of replication method you should choose and how to configure it depends on whether you have a legacy Swarm implementation and on your needs for securing replication traffic over untrusted networks. (v10.0)

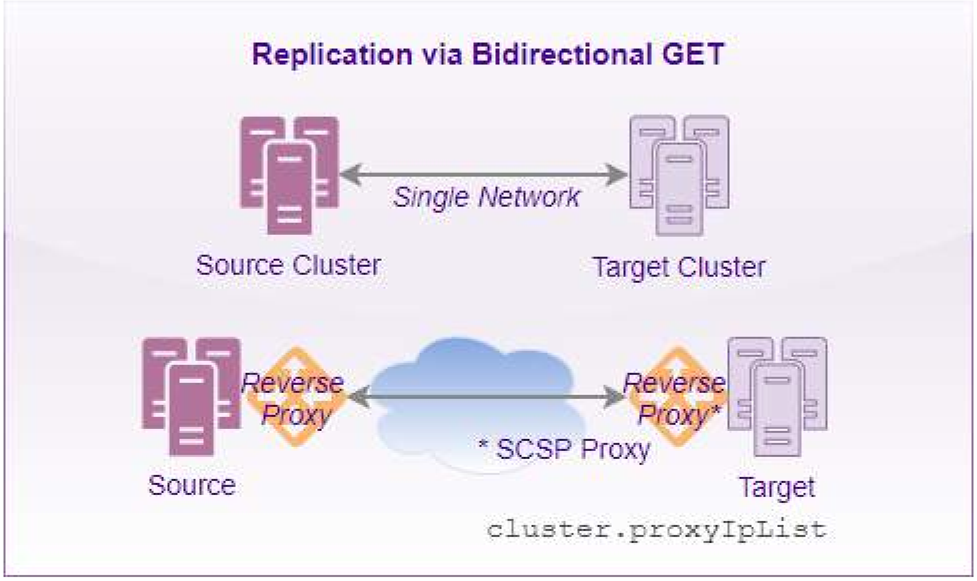
Secure Replication — Swarm Storage supports remote replication over a WAN, so that replication feeds can operate through Content Gateway. When you define a replication feed, you specify which replication mode to use: either the legacy bidirectional GET method of replication (which you may need for specific application compatibility or network requirements) or the recommended direct POST method, which offers better performance and flow management. With Swarm Storage 10.0 and later, you can implement TLS/SSL security as fits your implementation:

- Upload a trusted certificate to Swarm
- Replicate to an SSL offloader that services the target cluster
- Replicate from a forward proxy on your source cluster.

> See Replicating Feeds over Untrusted Networks and Adding a Trusted Certificate to Swarm.

Replication Options — Below are two replication methods available to you, along with the configuration variants of each that are supported:

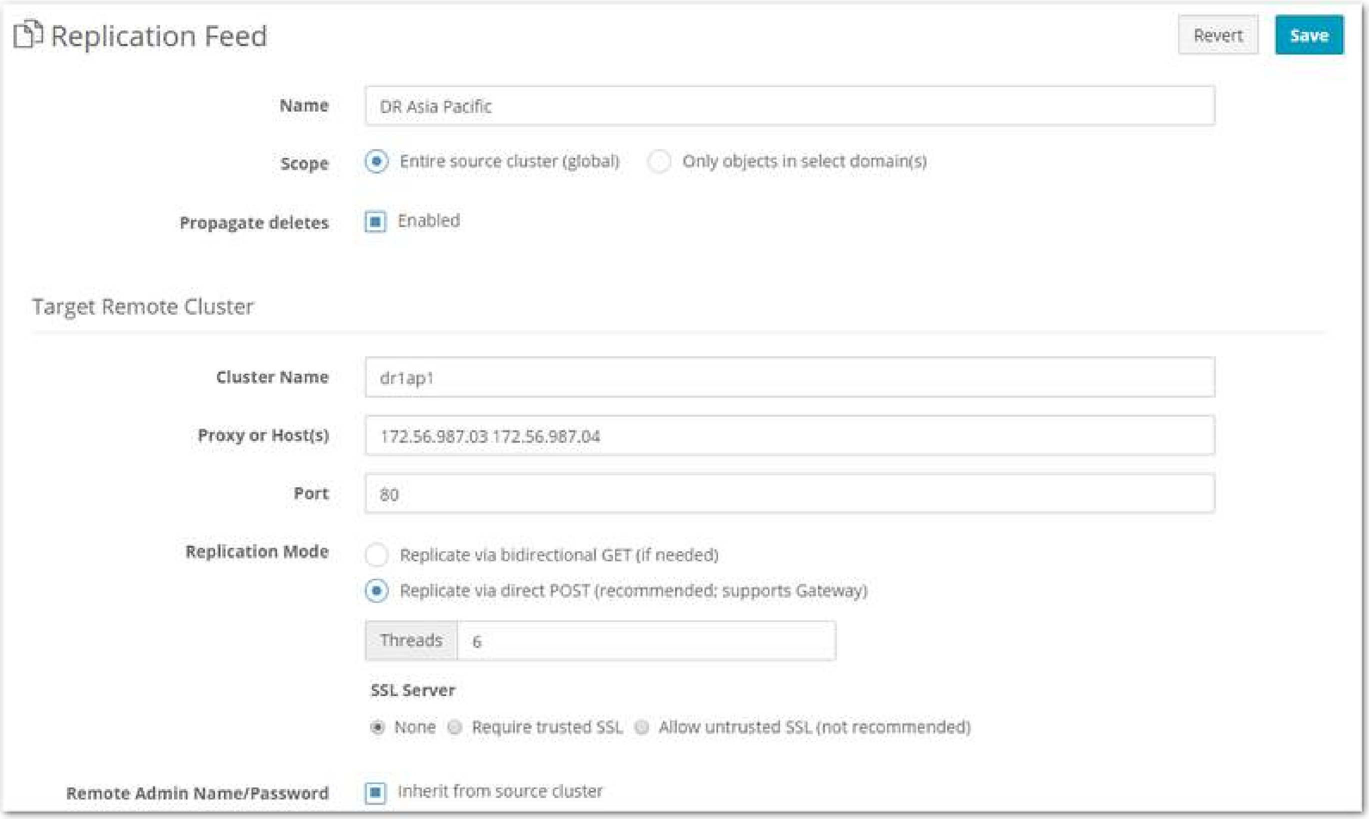




Note

Using Bidirectional GET for remote replication requires that you populate the Storage configuration setting `cluster.proxyIpList` for any cluster using a reverse proxy. The setting is a comma-separated list of reverse proxy IP addresses or names, including ports in `name:port` format. If using Direct POST replication, this setting can be populated or left blank, as it has no effect.

When you define a replication feed, set the scope and select which type (Replication Mode) is in force and with what speed (number of concurrent Threads), if you are using direct POST:



Name: DR Asia Pacific

Scope: ● Entire source cluster (global) ○ Only objects in select domain(s)

Propagate deletes: ☑ Enabled

Target Remote Cluster

Cluster Name: dr1ap1

Proxy or Host(s): 172.56.987.03 172.56.987.04

Port: 80

Replication Mode: ○ Replicate via bidirectional GET (if needed) ● Replicate via direct POST (recommended; supports Gateway)

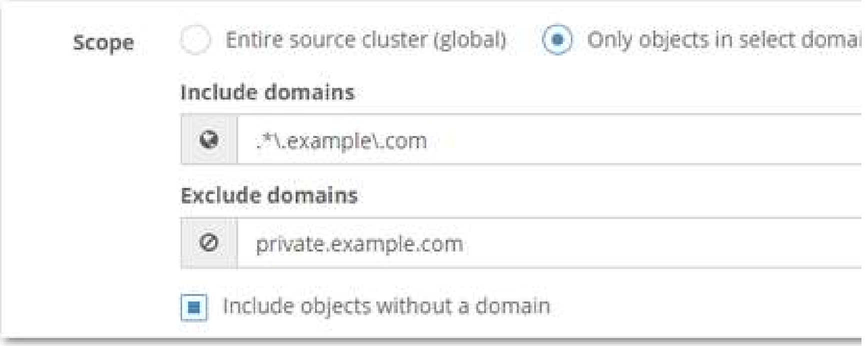
Threads: 6

SSL Server: ● None ○ Require trusted SSL ○ Allow untrusted SSL (not recommended)

Remote Admin Name/Password: ☑ Inherit from source cluster

The following table describes the data entry fields in the dialog box.

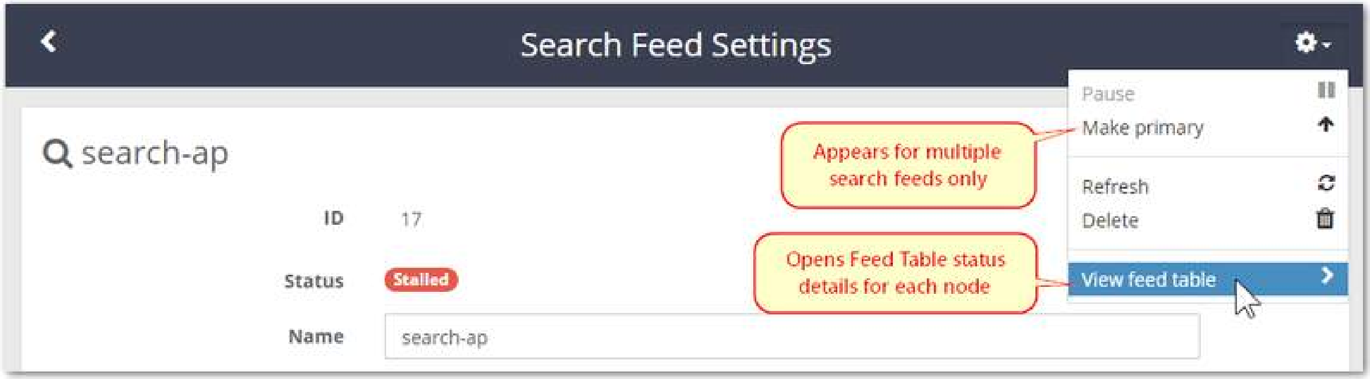
ID (existing feeds)		Read-only; system-assigned identifier
Status (existing feeds)	Active	Read-only; the current feed processing state. The state can be: • Active. Operating normally. • Recovering. Temporarily paused due to volume recovery. • Paused. Paused by user request. • Blocked. Processing blocked due to a transient condition. • Configuration error. Feed is unable to operate due to incorrect configuration • Overlapping feeds. More than the limit of 8 feeds have been defined for the s • Closed. Feed is inactive and no longer operational.
Name		The name you attach to this feed.

Scope	Entire source cluster (global) Only objects in select domain(s) Include objects without a domain	The feed filters that you select for your replication feed. The object will only be re you indicate in this field. • To replicate all objects in the source cluster, leave the default selection of En • To replicate only the objects in one or more domains, select the 'Only objects In the text box that appears, enter one or more domains: • To replicate only the objects within a specific domain, enter that domain. • To replicate only the objects within multiple domains, enter those domain and/or use pattern matching. • To exclude domains from replication, enter them. (v10.0) The field value allows pattern matching with the the Python regular express domain names can be matched. The exception to the RE matching is that th may not be used. An example domain list value using RE is: `.*\.example\.com` This would match both of these domains: `accounting.example.com, eng`  Scope ○ Entire source cluster (global) ● Only objects in select domai Include domains .*\.example\.com Exclude domains private.example.com ☑ Include objects without a domain • To replicate any unnamed objects that are not tenanted in any domain, enabl
Propagate deletes	Enabled	Select this checkbox if deletes should be propagated to the remote cluster.
Target Remote Cluster Name		The configuration setting for your target cluster (for example, the `cluster.name` the target cluster).
Proxy or Host(s)		The IP address(es) or host name(s) of either: • One or more nodes in the target cluster. • A reverse proxy host that routes to the target cluster. To enter two or more node IP addresses, enter each address separated by a com
Port	80	Lets you specify a custom port for the remote cluster.
Replication Mode	Direct POST	Choose replication via direct POST (recommended) or bidirectional GET. Switchi feed restart. (v1.2) For best performance, choose direct POST replication, which can go through Gat legacy method, which may be needed for application compatibility or networking

Threads	6	Replication via direct POST only. The default replication speed (6 simultaneous tl same-sized clusters with minimal replication backlog. (v1.2) To avoid overwhelming a smaller target cluster, reduce the threads. For faster re increase the threads temporarily, but be sure to monitor bandwidth and cluster p speed will stress both clusters.
SSL Server	None	Replication via direct POST only. If you are replicating over an untrusted network, Allow untrusted SSL is available but not intended for production systems. (v2.0)
Remote Admin Name/Password	Inherit from source cluster: Enabled User/Password credentials	Uncheck the enabled box, only if the remote cluster user name is different from t the same realm. Then enter: ● The administrative user name of the target cluster. ● The administrative password of the target cluster.

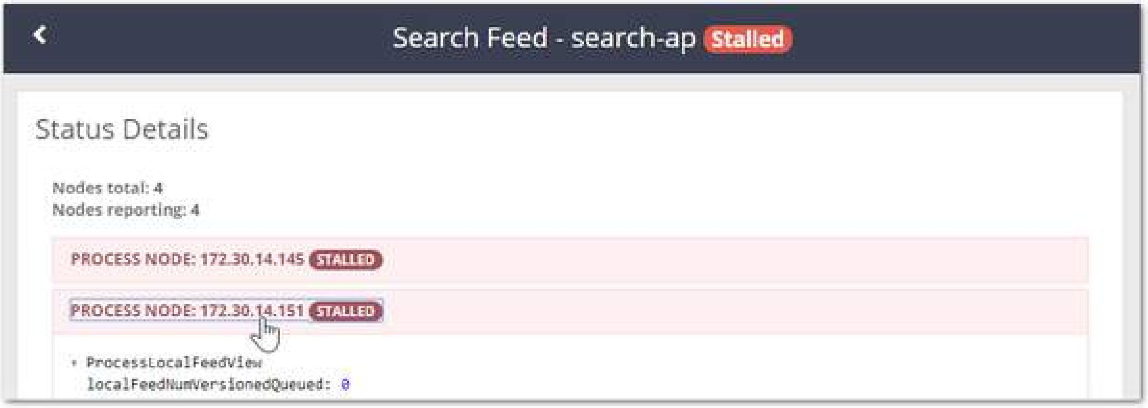
Managing Feeds

For an existing replication or search feed, clicking on it in the Feeds list will open its Feed Settings page, with the existing settings populated. The gear icon menu at the top right supports multiple feed actions.



Pause / Resume	You may occasionally wish to pause feed processing in order to perform system maintenance. For example, when you are upgrading your Elasticsearch cluster, you would pause the search feed before stopping the Elasticsearch service in your search cluster. After completing your system maintenance, return to the action menu and select the Resume action to resume feed processing.
Make Primary	For Search feeds only, select the 'Make primary' option from the feed actions menu to change which search feed is the primary feed that should be used for all search queries.

Refresh	As objects are written or updated, their data is sent to the feed target in near real-time (NRT). Any objects that cannot be processed immediately are retried each HP cycle until they succeed, at which point they are marked as complete and are never resent. If a data loss failure occurs on your remote feed target and you cannot restore from backup, select the Refresh option from the feed action menu, which will verify and rehydrate all of the previously sent content to either a remote cluster or your Elasticsearch cluster. This process will take some time as it must revisit all objects in the cluster. For search feeds, if an Elasticsearch index for the cluster does not exist, it will be created. To recreate an existing index 'fresh' (such as for case-insensitive searching where case-sensitive was previously used), drop the existing index before refreshing the feed.
Delete	When you delete a feed, it frees source cluster resources. This process does not affect the objects previously pushed to the remote target. To delete a feed, select the Delete option from the feed action menu and confirm you intend to permanently delete the feed. The deleted feed is removed from the remaining cluster nodes within 60 seconds. If you wish, you may also delete the search data previously sent by the feed.

View feed table	To troubleshoot blocked feeds, select View feed table, which displays the the SNMP Repository Dump for the selected node. (v2.0)
	
	Search Feed - search-ap Stalled
	Status Details
	Nodes total: 4
	Nodes reporting: 4
	PROCESS NODE: 172.30.14.145 STALLED
	PROCESS NODE: 172.30.14.151 STALLED
	`▸ ProcessLocalFeedView`
	`localFeedNumVersionedQueued: 0`
	Review the feedPluginState status to identify the blockage.
	Example: `feedPluginState blocked: Destination cluster onyx1 reports invalid request: Castor-System-Cluster value must refer to a remote cluster on RETRIEVE request`

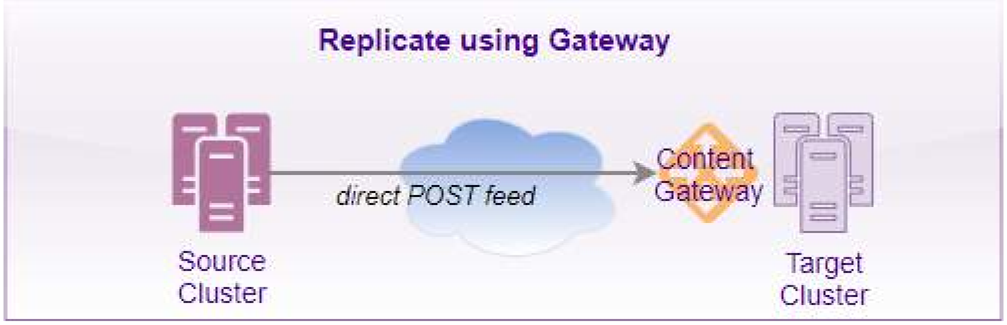
Upgrades from Swarm 8

For replication feeds, a Swarm 8 bug replicated all domain and bucket objects (not their contents), even if the feed restricted replication to certain domains. If you used mirrored replication with Swarm 8, do not delete the erroneous domain and bucket objects from the destination cluster; contact Support for assistance. (SWAR-7100)

- Replicating Feeds over Untrusted Networks
- Adding a Trusted Certificate to Swarm

Replicating Feeds over Untrusted Networks

With Content Gateway 5.4 and higher, you can create replication feeds using direct POST replication, with Gateway proxying the target cluster.



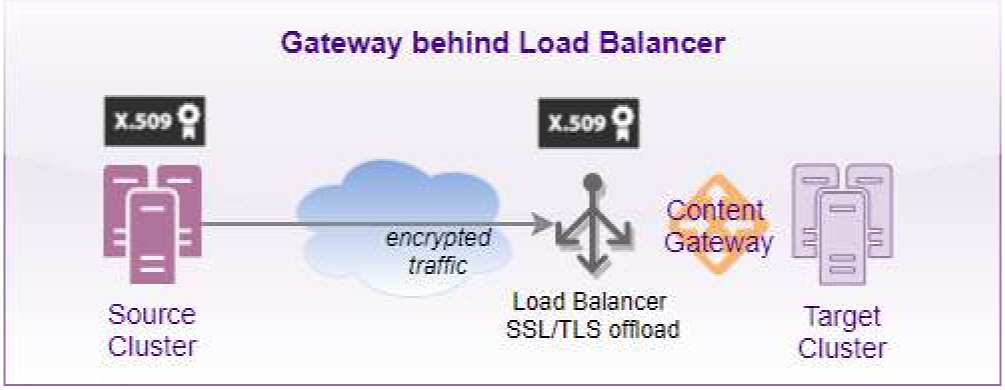
Note
Gateway's `cluster.proxyIPList` setting is for use with legacy bidirectional GET replication only. See Viewing and Editing Feeds.

However, if your source cluster needs to replicate feeds over an untrusted network securely with SSL/TLS encryption, you can add a trusted certificate and a third-party proxy to handle redirects.
For more on creating a self-signed SSL certificate, see Adding a Trusted Certificate to Swarm.

- Load Balancer (Offloader)
    - Setting Up an Offloader
    - Configuring the Feed
- Forward Proxy
    - Choosing a Forward Proxy Server
    - Reconfiguring the Feed

Load Balancer (Offloader)

If you use a load balancer for SSL/TLS offload (which is a type of reverse proxy), you will need to configure the source Swarm cluster and your load balancer to use a trusted connection:



Setting Up an Offloader

> Important
>
> The ports specified in the proxy configuration must match the bind ports specified in the Gateway Configuration.

This example shows how to configure haproxy as an SSL offloader for Content Gateway on RHEL/CentOS 7.

1. Check the Content Gateway configuration and note which ports are being used for SCSP and S3. These ports must match in your offloader's setup.

   /etc/caringo/cloudgateway/gateway.cfg

   ```
   [scsp]
   enabled = true
   bindAddress = 0.0.0.0
   bindPort = 8080
   externalHTTPport = 443

   [s3]
   enabled = true
   bindAddress = 0.0.0.0
   bindPort = 8090

   [cluster_admin]
   enabled = true
   bindAddress = 0.0.0.0
   bindPort = 91
   externalHTTPSport = 91
   ```

   > Forcing HTTPS
   >
   > This configuration still provides HTTP access; to harden your security and force HTTPS, change all of the `bindAddress` settings to 127.0.0.1.

2. Setup and install haproxy. This package is part of the EPEL repo.
3. Use the following haproxy configuration:

   /etc/haproxy/haproxy.cfg

   ```
   global
       log 127.0.0.1 local2
       chroot /var/lib/haproxy
       stats socket /var/lib/haproxy/stats mode 660 level admin
   ```

```
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

    ca-base /etc/pki/tls/certs
    crt-base /etc/pki/tls/private

    ssl-default-bind-ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH
+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS
    ssl-default-bind-options no-sslv3
    maxconn 2048
    tune.ssl.default-dh-param 2048

defaults
    log     global
    mode    http
    option  forwardfor
    option  http-server-close
    option  httplog
    option  dontlognull
    timeout connect 5000
    timeout client  50000
    timeout server  50000

frontend www-http
    bind 0.0.0.0:80
    reqadd X-Forwarded-Proto:\ http
    reqadd X-Forwarded-Port:\ 80
    default_backend www-backend-scsp
    acl iss3 hdr_sub(Authorization) AWS
    acl iss3 url_reg [?&](AWSAccessKeyId|X-Amz-Credential)=
    use_backend www-backend-s3 if iss3

frontend www-https
    bind 0.0.0.0:443 ssl crt /etc/pki/tls/certs/selfsignedcert.pem
    reqadd X-Forwarded-Proto:\ https
    reqadd X-Forwarded-Port:\ 443
    default_backend www-backend-scsp
    acl iss3 hdr_sub(Authorization) AWS
    acl iss3 url_reg [?&](AWSAccessKeyId|X-Amz-Credential)=
    use_backend www-backend-s3 if iss3

frontend www-https-svc
```

```
    bind 0.0.0.0:91 ssl crt /etc/pki/tls/certs/selfsignedcert.pem
    reqadd X-Forwarded-Proto:\ https
    reqadd X-Forwarded-Port:\ 91
    default_backend www-backend-svc

backend www-backend-scsp
    #redirect scheme https if !{ ssl_fc }   <--- Uncomment to force
HTTPS
    server gw1 127.0.0.1:8080 check

backend www-backend-s3
    #redirect scheme https if !{ ssl_fc }    <--- Uncomment to
force HTTPS
    server gw1 127.0.0.1:8090 check

backend www-backend-svc
    # This rule rewrites CORS header to add the port number used on
frontend
    http-request replace-value Access-Control-Allow-Origin (.*)
```

```
\1:91
    redirect scheme https if !{ ssl_fc }
    server gw1 127.0.0.1:8091 check
```

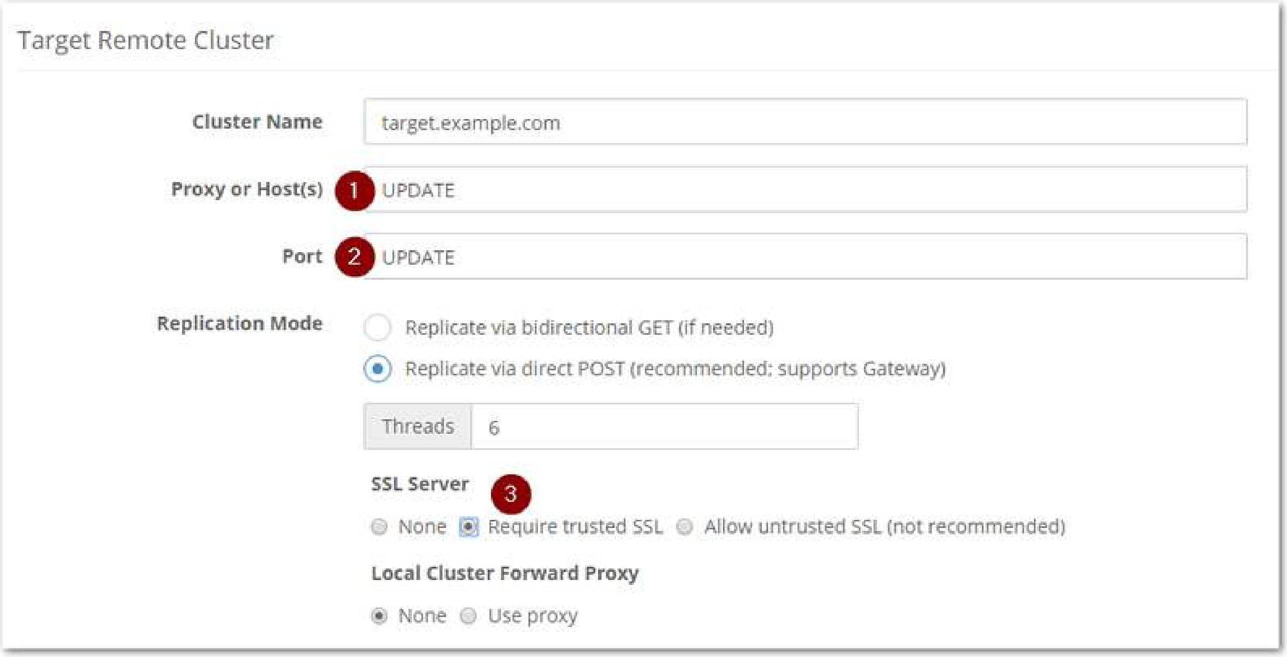
4. Start haproxy.

```
systemctl restart haproxy
```

Configuring the Feed

Once you have set up your SSL server, you need to configure the Swarm replication feed to use it.

1. In the Swarm UI, go to Cluster > Feeds.
2. Edit the affected replication feed.
3. Scroll to the Target Remote Cluster settings.
4. Update the Proxy or Host(s) and Port to point to your offloader.
5. If the feed was configured to use the bidirectional GET mode, select Replicate via direct POST.
6. For SSL Server, enable Require trusted SSL; Allow untrusted SSL is available but not intended for production systems.



Target Remote Cluster

Cluster Name: target.example.com

Proxy or Host(s): UPDATE

Port: UPDATE

Replication Mode:
○ Replicate via bidirectional GET (if needed)
● Replicate via direct POST (recommended; supports Gateway)

Threads: 6

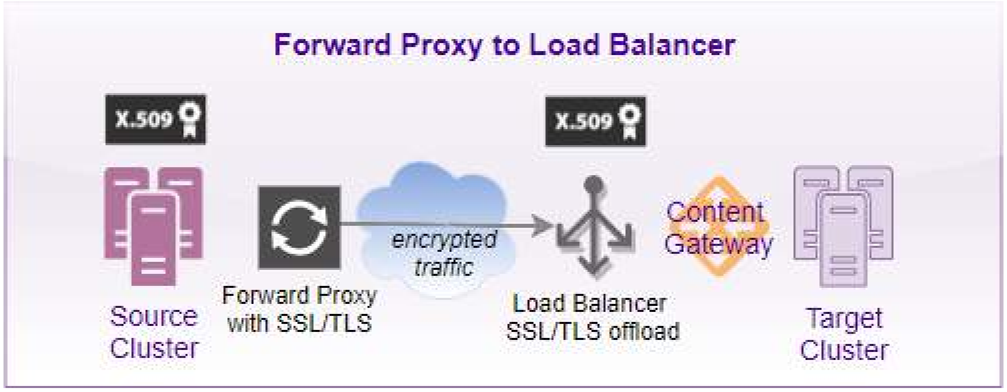
SSL Server
○ None ● Require trusted SSL ○ Allow untrusted SSL (not recommended)

Local Cluster Forward Proxy
● None ○ Use proxy

7. Select None for Local Cluster Forward Proxy, unless you are using one (see Forward Proxy, below).

> See Viewing and Editing Feeds.

Forward Proxy


Forward Proxy to Load Balancer
X.509
X.509
Source Cluster
Forward Proxy with SSL/TLS
encrypted traffic
Load Balancer SSL/TLS offload
Content Gateway
Target Cluster

Choosing a Forward Proxy Server

Although HAProxy is not optimized to be a general purpose forward proxy, it should work with a fixed back-end server list that consists of the distant Gateway front-end.

Other alternatives:

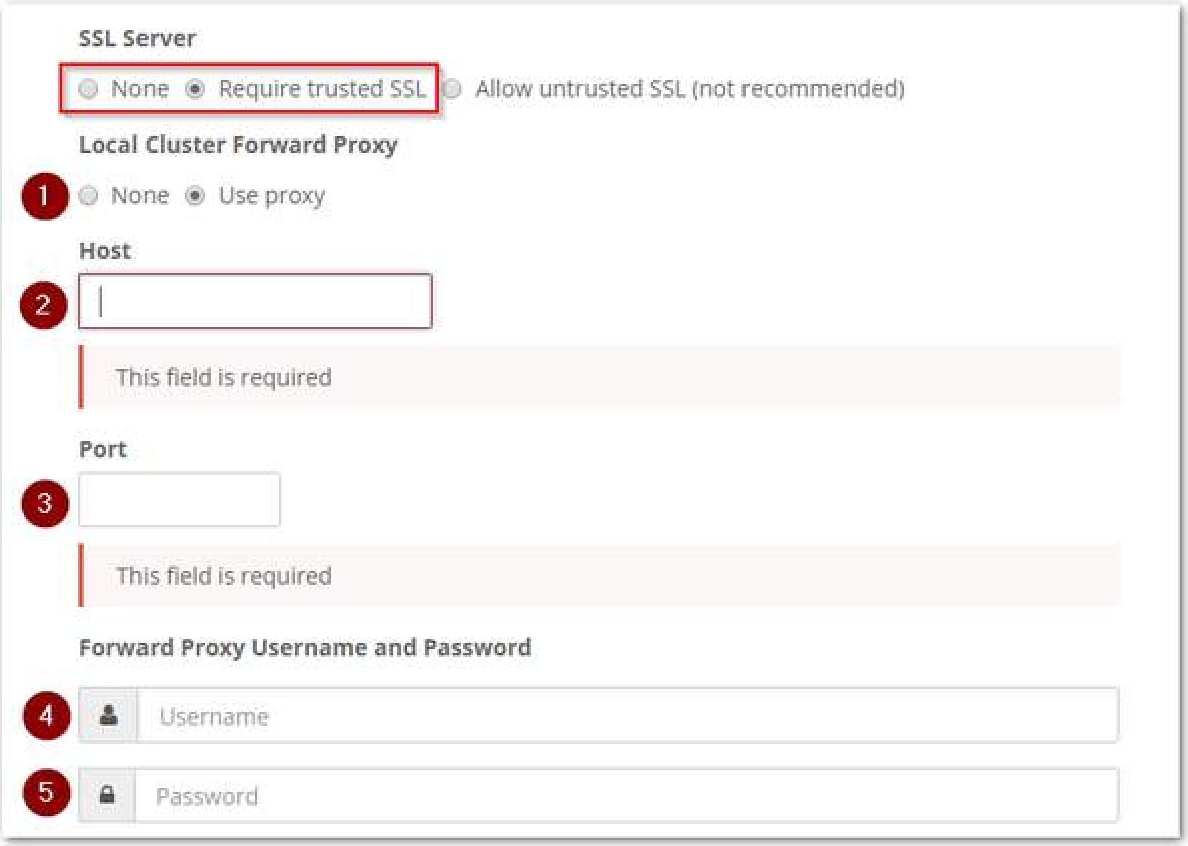
- stunnel — for fixed endpoints
- Squid — for a general purpose forward proxy

With this server configuration, your forward proxy receives an HTTP request from the Swarm node and then tunnels a Swarm HTTPS request over the Internet to the other cluster, hitting the SSL/TLS offloader in front of Gateway. The data is encrypted by Swarm, and passed blindly through the forward proxy.

Reconfiguring the Feed

Once you have set up a forward proxy server, you need to then configure the Swarm replication feed to use your new outbound proxy.

1. In the Swarm UI, go to Cluster > Feeds.
2. Edit the replication feed that you already configured to use an SSL Server.
3. For Local Cluster Forward Proxy, select Use proxy.
4. Enter the Host (a fully qualified domain name or IP address) and the Port for the proxy.
5. Enter the Username and Password for the forward proxy.



SSL Server

○ None ● Require trusted SSL ○ Allow untrusted SSL (not recommended)

Local Cluster Forward Proxy

○ None ● Use proxy

Host

This field is required

Port

This field is required

Forward Proxy Username and Password

Username

Password

> See Viewing and Editing Feeds.

Adding a Trusted Certificate to Swarm

Certificate Essentials

If your Swarm site collects or transmits personally identifiable information or otherwise needs to protect its traffic, you need to add an X.509 security certificate (also called an "SSL Certificate") so that Swarm can be trusted for a secure connection. TLS (Transport Layer Security) or SSL (Secure Sockets Layer) security is made up of two parts:

1. Encryption — data is made unreadable (using an encryption key) and then sent over an HTTPS connection (SSL), and it can only be read by a client that has the needed key.
2. Identification — transmission is certified (with a security certificate) as coming from the authentic (trusted) site.

You have two options for certificates:

- Pay a trusted CA (Certificate Authority, such as Verisign) to approve (sign) a certificate: "Trust me, because Verisign vouches that I am who I claim to be." (Typically needed for e-commerce.)
- Create your own self-signed certificate: "Trust me, I am who I claim to be."

Both certificate types will encrypt the data to create a secure website that cannot be read by third-parties.

> Note
> Be aware that most browsers check whether an HTTPS connection is signed by a recognized CA. If the connection is self-signed, it could be flagged as potentially risky, even though it is secure.

Making a Self-Signed SSL Certificate

> X.509 Required
> Swarm requires that SSL certificates use X.509 (`openssl req -x509`), which is a public key infrastructure standard that SSL and TLS adhere to for key and certificate management.

There are many ways and tools for creating trusted certificates. Here is one way for making a self-signed certificate to add to a proxy that you might use in front of a Swarm cluster:

1. Set up a secure (root-only access) directory for holding the private key and certificate files.
2. Generate a unique private key (KEY).

   ```
   $ openssl genrsa -out mydomain.key 2048
   ```

   File contents start with: `-----BEGIN RSA PRIVATE KEY-----`
3. Generate a Certificate Signing Request (CSR).

   ```
   $ openssl req -new -key mydomain.key -out mydomain.csr
   ```

   File contents start with: `-----BEGIN CERTIFICATE REQUEST-----`
4. Create a self-signed certificate (CRT), filling out the `openssl` prompts appropriately (most importantly, the **`Common Name`**, which may be a domain name or public IP address).

   ```
   $ openssl x509 -req -days 365 -in mydomain.csr -signkey
   mydomain.key -out mydomain.crt
   ```

   File contents start with: `-----BEGIN CERTIFICATE-----`

   > S3 wildcards
   > If you use S3, make a wildcard SSL certificate as well: run the command again, but when prompted for **`Common Name`**, use a wildcard: `*.DOMAIN`

5. Append KEY and CRT(s) to mydomain.pem.

   ```
   $ bash -c 'cat mydomain.key mydomain.crt mydomainwildcard.crt >>
   /etc/pki/tls/certs/mydomain.pem
   ```

   PEM file gains multiple sections: `-----BEGIN RSA PRIVATE KEY-----` and `-----BEGIN CERTIFICATE-----`
6. Specify the PEM in the configuration file of the proxy, such as HAProxy.

   ```
   $ vim /etc/haproxy/haproxy.cfg
   ```

   - Locate: `bind 0.0.0.0:443 ssl crt /etc/pki/tls/certs/selfsignedcert.pem`
   - Update to: `bind 0.0.0.0:443 ssl crt /etc/pki/tls/certs/selfsignedcert.pem crt /etc/pki/tls/certs/mydomain.pem`
7. Restart the proxy.

   ```
   $ service haproxy restart
   ```

Uploading Certificates into Swarm

To protect Swarm traffic over untrusted networks, you need to upload your trusted certificate (public key) to Swarm, which you do by inserting it into Swarm's settings. The `startup.certificates` setting holds any and all certificates, formatted as a single line.

> Tip
> You can concatenate multiple certificates into one.

Until the Swarm UI implements upload support for certificates, you can prepare and upload certificates by hand. The setting is read-only. Because it requires editing of the configuration file, activating the certificate requires a reboot.

1. Modify your PEM certificate(s) so that the key is a single line, with all of the carriage returns replaced with the newline character: `\n`.
   The following awk command converts a PEM file into the string that you need:

   ```
   awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' mydomain.pem
   ```

2. Edit your Swarm configuration file (node.cfg, cluster.cfg).
3. At the end of your file, insert the certificates setting:

```
[startup]
certificates=
```

4. Directly after the equal sign, paste in your single-line string (shown here abridged), which contains all of the needed newline characters (\n):

```
[startup]
certificates=----BEGIN CERTIFICATE----- {snip} ----END
CERTIFICATE-----
```

5. Reboot the node. If Swarm does not boot, the value provided may not be a valid x.509 public certificate, so check the formatting.

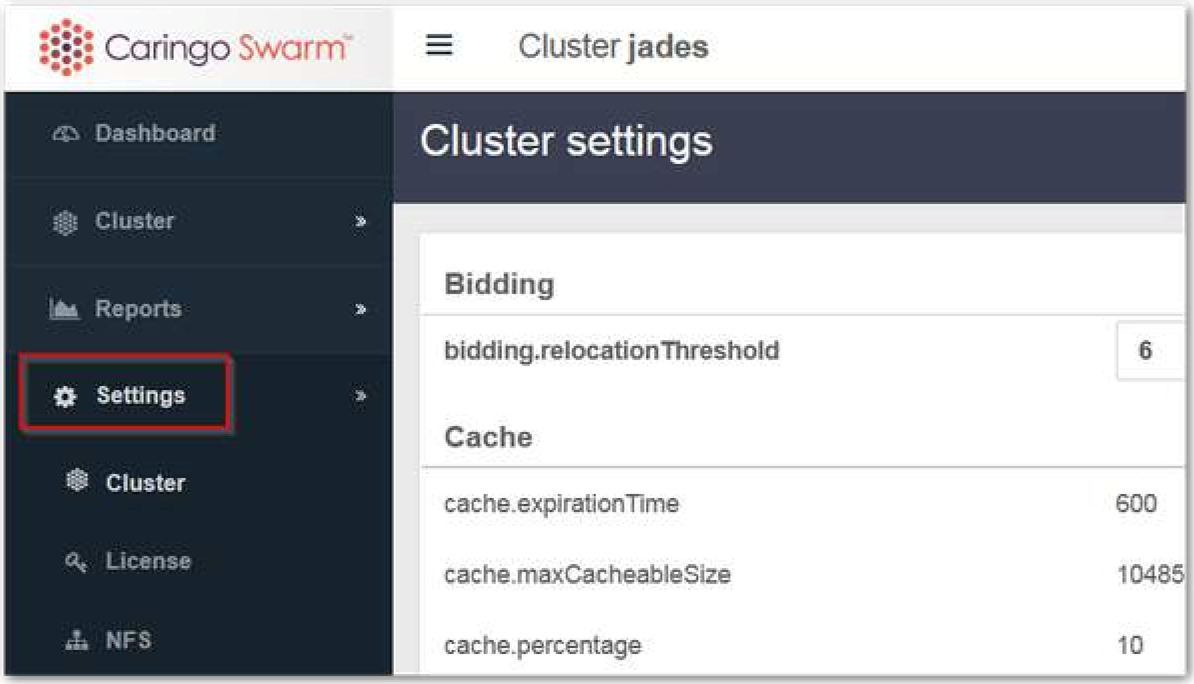
When Swarm reboots, it appends the key value to `/etc/ssl/certs/ca-certificates.crt`.

Using Cluster Settings

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Cluster Settings
- License Settings

The Settings section includes cluster-wide settings, license information, and optional NFS configuration.



Cluster settings

Bidding

bidding.relocationThreshold 6

Cache

cache.expirationTime 600

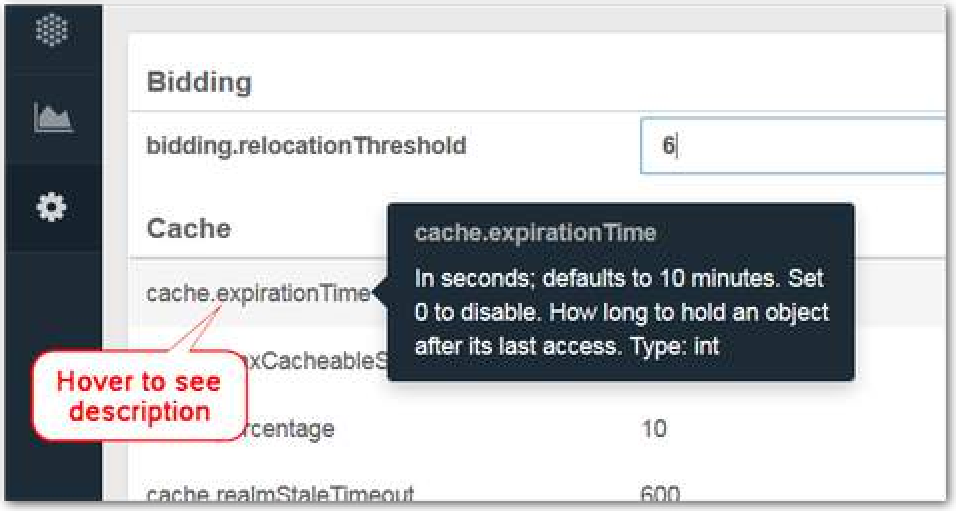
cache.maxCacheableSize 10485

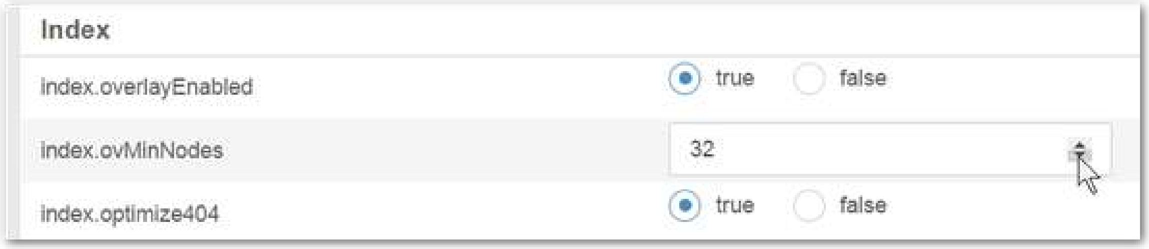
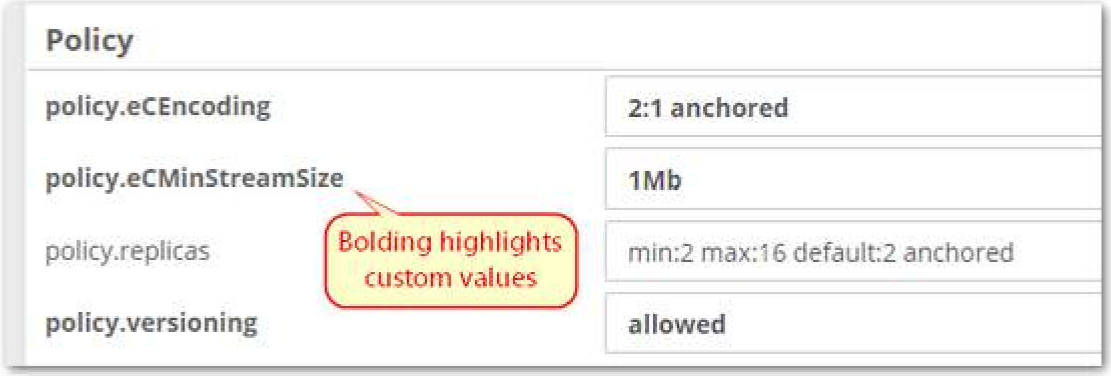
cache.percentage 10

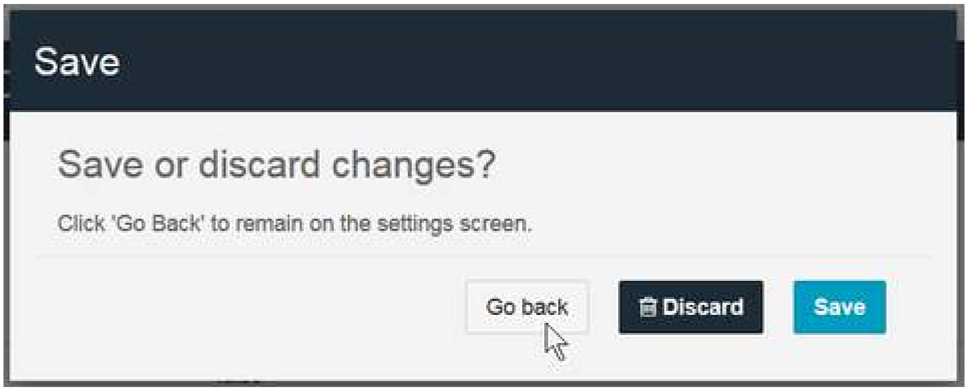
> See SwarmNFS Export Configuration.

Cluster Settings

Swarm Storage supports a variety of settings for various purposes. All possible settings for the storage cluster are displayed on the Cluster settings page with their current values.

Descriptions	Mouse over the setting name to see the setting description; you can also refer to the Settings Referenc for complete details.  Bidding bidding.relocationThreshold 6 Cache cache.expirationTime cache.expirationTime In seconds; defaults to 10 minutes. Set 0 to disable. How long to hold an object after its last access. Type: int xCacheableS rcentage 10 cache.realmStaleTimeout 600

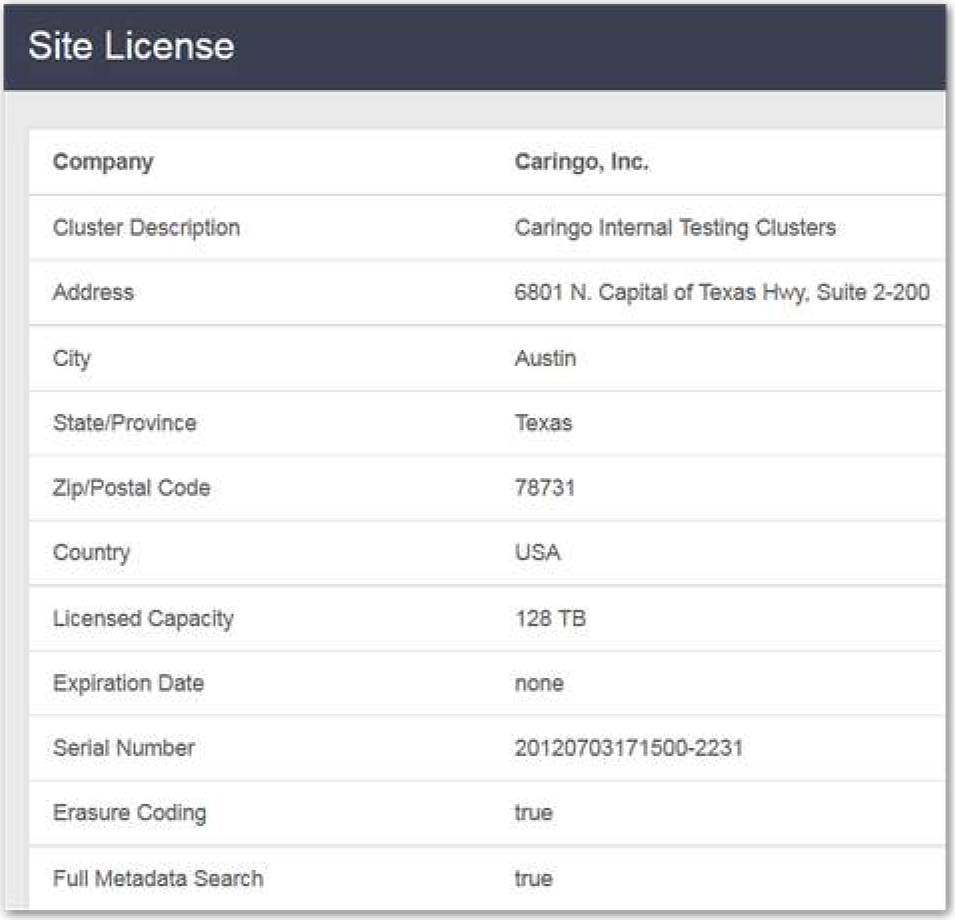
Writable values	Settings that can be updated on a running cluster display with an edit box or radio toggle to allow for quick editing.  Index index.overlayEnabled ● true ○ false index.ovMinNodes 32 index.optimize404 ● true ○ false Note To edit settings that do not support dynamic updates (those that are not persisted settings), you must update your cluster configuration file (on the CSN or wherever it resides) and reboot all storage chassis in order for the new setting to be applied.
Custom values	For easy identification, all customized settings appear in bold. This does not apply to settings that have no default.  Policy **policy.eCEncoding** **2:1 anchored** **policy.eCMinStreamSize** **1Mb** policy.replicas min:2 max:16 default:2 anchored **policy.versioning** **allowed** Bolding highlights custom values

Changing values	Changing any editable settings will enable the Save button in the upper right corner. Any in process changes can be returned to their previous saved value by using the Revert button. If you navigate away from the page with unsaved changes, the page prompts you to decide what to do with those edits:  Save Save or discard changes? Click 'Go Back' to remain on the settings screen. Once you Save settings changes, they propagate in the cluster within 60 seconds.

License Settings

The license information for the license that is currently in use by the cluster displays on the License Settings page. Scroll down to see the date that it was last updated.

> Note
>
> The License Settings page requires Platform Server in order to update the license from the Swarm UI; otherwise, it is read-only.



Site License

Company	Caringo, Inc.
Cluster Description	Caringo Internal Testing Clusters
Address	6801 N. Capital of Texas Hwy, Suite 2-200
City	Austin
State/Province	Texas
Zip/Postal Code	78731
Country	USA
Licensed Capacity	128 TB
Expiration Date	none
Serial Number	20120703171500-2231
Erasure Coding	true
Full Metadata Search	true

Configuring SwarmNFS

- Adding Server Groups
- Adding Exports
  - Client Access
  - Cloud Security
  - Advanced Settings
- Mounting the Exports

The NFS page in the Swarm Storage UI lets you create and manage your NFS server groups and exports.

> Important
>
> You must create the storage cluster's default domain before configuring SwarmNFS. This domain has the same name as the `cluster.name` setting's value. The domain can be created with the Content UI or an HTTP utility like curl.

You can create separate groups (sets) of SwarmNFS that are configured in pools; this lets you support different clients and optimize for different roles. You can also set some configuration settings locally, to override global configuration settings.

Why different server groups? These are typical configuration changes that you might want to make for specific servers:

- Include DEBUG level logging
- Change the log file location
- Add local resource restrictions
- Change interface or IP address bindings
- Reduce maximum threads or open/concurrent client connections

While every SwarmNFS server will retrieve the global configuration file stored within Swarm, each server group can optionally override the global settings with their own configuration file.

> Important
> Restart NFS services after making any configuration changes. The NFS server does not yet support dynamic updates to the running configuration.

Adding Server Groups

Server Groups are created with the + Add button at top right.

> Best practice
> Before creating a Server Group, be sure that your default domain is specified and to verify the existence of the domain and bucket that will be defined in the scope.

The resulting group is a container for exports that will share a common configuration:

Name	When you add a Server Group, you only supply a name, which is a description; the unique identifier is the count at the end of the Configuration URL. The new group appears at or near the end of the listing, ready to be configured with exports.
Configuration URL	Each NFS Server Group has a unique Configuration URL, which you can click to view the current export definitions. These are the auto-generated and auto-maintained JSON settings being stored by Swarm for the group. Note The configuration is empty until you add one or more exports.

> Important
> Although group configurations may be shared across NFS servers, each server must be configured with only one group.

Adding Exports

Each SwarmNFS export that you create with +Add Export goes direct to Swarm and uses anonymous/guest authentication.

> Note
>
> Each export is specific to one and only one Swarm bucket, but clients viewing the mounted directory will be able to view, create, and use virtual directories within it via the prefix feature of Swarm named objects (`myvirtualdirectory/myobjectname.jpg`).

Name		Unique name for your export, to distinguish it from the others in Swarm UI listings.
Storage IP(s) or DNS name(s)		The IP address(es) or DNS-resolvable hostname(s) for one or more Swarm Storage nodes.
Search host(s)		The IP address(es) or DNS-resolvable hostname(s) for one or more Swarm Elasticsearch servers. Note: Both Gateway and SwarmNFS use the Primary (default) search feed. If a new feed is made Primary, these servers must be restarted.
Search index		Optional, but recommended for larger sites to improve performance. The unique alias name of the primary (default) search feed. See below for how to identify the value.
Export path		Case-sensitive. Unique pseudo filesystem path for the NFS export. Cannot be set to a single slash ("`/`"), which is reserved.
Scope	Domain Bucket	Specifies where the data written via the export will be associated: which domain and bucket to use. Important: Be sure to verify the existence of the domain and bucket that you specify here.

Storage interface	Gateway	When you deploy in a Gateway environment, you can use pass-through authentication, which means authenticating to Gateway using the same login and password that was provided for authentication by the client to SwarmNFS. You also have the choice of session tokens (with various expirations) and single user authentication, by login credentials or token. SwarmNFS maintains exactly the same level of object security when accessing or modifying objects through SwarmNFS or other protocol such as SCSP, S3 or the SwarmFS (Hadoop). Gateway provides security at domain and bucket level only and objects inherit those security policies, accessibility to all unnamed objects are restricted to that of the user's rights at the containing domain, and restricted to rights set at the containing bucket level for named objects. SwarmNFS layers no individual object security (named or unnamed) above that enforceable by Gateway.
	Direct to Swarm	When you deploy without Gateway, SwarmNFS uses anonymous/guest authentication.

> See Identifying the Primary Search Feed.

Client Access

This optional section allows you to customize access control both globally (for this export) and for specific clients.

Access type	Defaults to full read/write access. These other access restrictions are available: • All operations (RW) - default • No access (None) • Read-only (RO) • No read/write (MDONLY) - allows listing and metadata updates without access to file contents • No read/write/modify (MDONLY_RO) - allows listing but no metadata updates and no access to file contents
Squash	Defaults to no squashing (allows all user IDs). • None - default • Root - squashes the remote superuser (root, uid=0) when using identity authentication (local user is the same as remote user) • All - squashes every remote user, including root.

Squash user id (uid) mapping Squash id (uid) mapping	User ID and Group ID can be set when you have the NFS server authenticating users from a different authentication sources and/or you want all the files to have a consistent user/group. Typical situations: • All clients are configured to use local password/group files, but SwarmNFS through the Content Gateway is configured to use LDAP. • All clients have local password/group files, but some users may not exist on all clients systems or may differ on each client. • All clients have the same users and groups, but they were created in a different order. • All clients authenticate using individual logins/accounts, but you want all files to have the same consistent owner and group regardless of the user reading or writing the files. • A client mounts the NFS exports as anonymous, but you want the files presented over the share to all NFS clients to have a consistent UID and GID.
Client(s)	As needed, customize the access for one or more specific clients. Note: These override the settings specified above, if any.

Cloud Security

Each SwarmNFS export that you create to use the Content Gateway can have a different security method, as needed by its use case:

Session Token	Token Admin Credentials by Login Token Admin Credentials by Token	User, Password, Expiration Token, Expiration
Single User	Authenticate by Login Authenticate by Token	User, Password Token
Pass-through / None	n/a	

Advanced Settings

> Important
> Use these recommended defaults for all of the Advanced Settings unless otherwise advised by Caringo Support.

Transport protocol	TCP	Supported transport protocol (TCP/UDP | TCP | UDP)
Storage port	80	Network port for traffic to Swarm Storage nodes
Search port	9200	Network port for traffic to Swarm Search nodes
Security	sys	Remote Procedure Call (RPC) security type (sys | krb5 | krb5i | krb5p)

Maximum storage connections	100	Maximum number of open connections to Swarm Storage. (v2.0)
Retries	5	(positive integer) How many times SwarmNFS will retry unsuccessful requests to Swarm and Swarm Search before giving up.
Retries timeout	30	(seconds) How long SwarmNFS will wait before timing out Swarm retries.
Request timeout	60	(seconds) How long SwarmNFS will wait before timing out Swarm requests. For best results, set this timeout to at least twice the value of the Storage setting scsp .keepAliveInterval.
Pool timeout	300	(seconds) How long discovered Swarm storage nodes are remembered.
Write timeout	60	(seconds) How long SwarmNFS will wait for a write to Swarm to complete before retrying.
Read buffer size	16000000	(bytes) The amount of data to be read each time from Swarm. If the read size buffer is larger than the client request size, then the difference will be cached by SwarmNFS, and the next client read request will be served directly from cache, if possible. Set to 0 to disable read ahead buffering
Maximum buffer memory	2000000000	(bytes) Defaults to 2GB. Maximum limit that can be allocated for the export's export buffer pool. Once exceeded, client requests will temporary be blocked until total buffers falls back below this number. (v2.0)
Buffer high watermark	1500000000	(bytes) Once the allocated export buffers reach this watermark, SwarmNFS will start to free buffers in an attempt to stay below “Maximum Memory Buffers”. During this time, client requests may be delayed. (v2.0)
File access time policy	"relatime"	Policy for when to update a file's access time stamp (atime). (v2.0) • “noatime”: Disables atime updates. • “relatime”: Updates atime only if it is earlier than last modified time, so that it updates only once after each write. • “strictatime”: Updates atime on every read and close.

Mounting the Exports

When mounting your SwarmNFS exports, follow these guidelines:

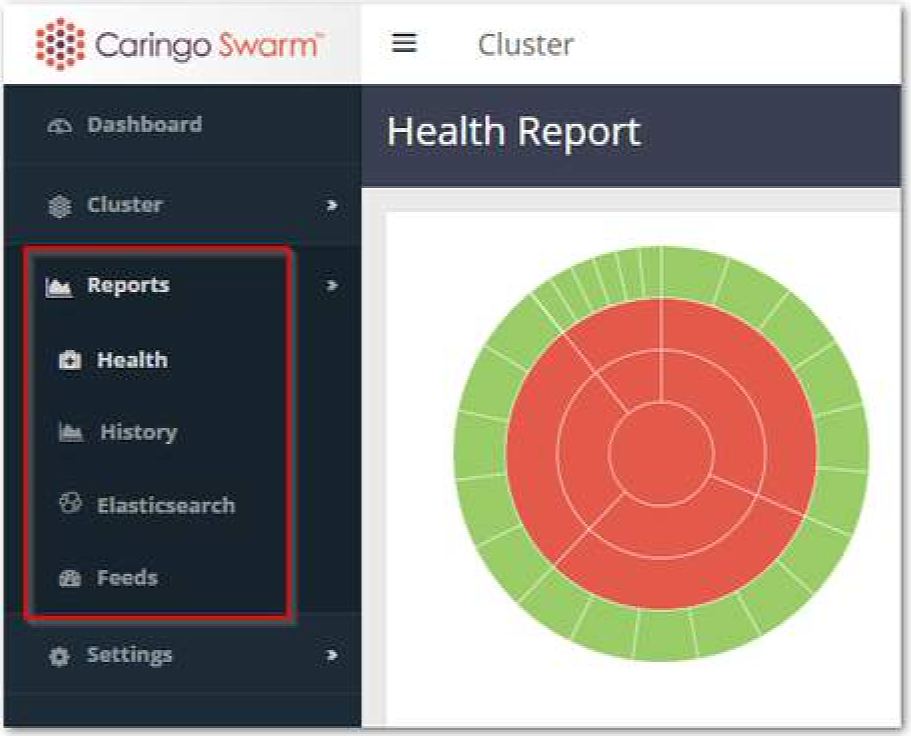
Linux	Mount the exports as normal, with these performance additions: - Enable the noatime (no access time) setting on your NFS export mounts, which lets the system skip updating the file system for files that are simply being read. Note that the write time information to a file will always be updated any time the file is written to. - Increase the timeout, timeo, to 9000. `mount -o timeo=9000,noatime SwarmNFSserver:/ /mnt/SwarmNFS`
OS X	Enable NFS 4.1 mounting.
Windows	You cannot mount SwarmNFS to Windows because it has no NFS 4.x client.

Note
SwarmNFS has responsive throttling to manage out-of-memory situations for your exports. This throttling will appear in clients as slower or paused activity until memory is freed up again. The syslog will contain warning messages alerting to the situation.

Using Cluster Reports

The Reports section of the Swarm UI includes valuable real-time and historical views into the health of and activity in your cluster.

- Health Report
- History Reports
- Elasticsearch Reports
- Feeds Reports



Health Report

The Health Report provides both summary and detail information at the level of cluster, subcluster, chassis, and drive.

The sunburst graphic shows an interactive visualization of the cluster, with the cluster itself represented in the center, subclusters in the next concentric circle out, chassis in the next concentric circle, and drives on the outside. To see only the data for a particular subcluster or chassis, click on its wedge in the sunburst. All summary and detail data will update to show only the selected component. To return to a higher level view, click the center of the sunburst. To make it easier to identify a component, the components id (IP Address, drive name, etc) and status will display as you mouse over each wedge in the sunburst.

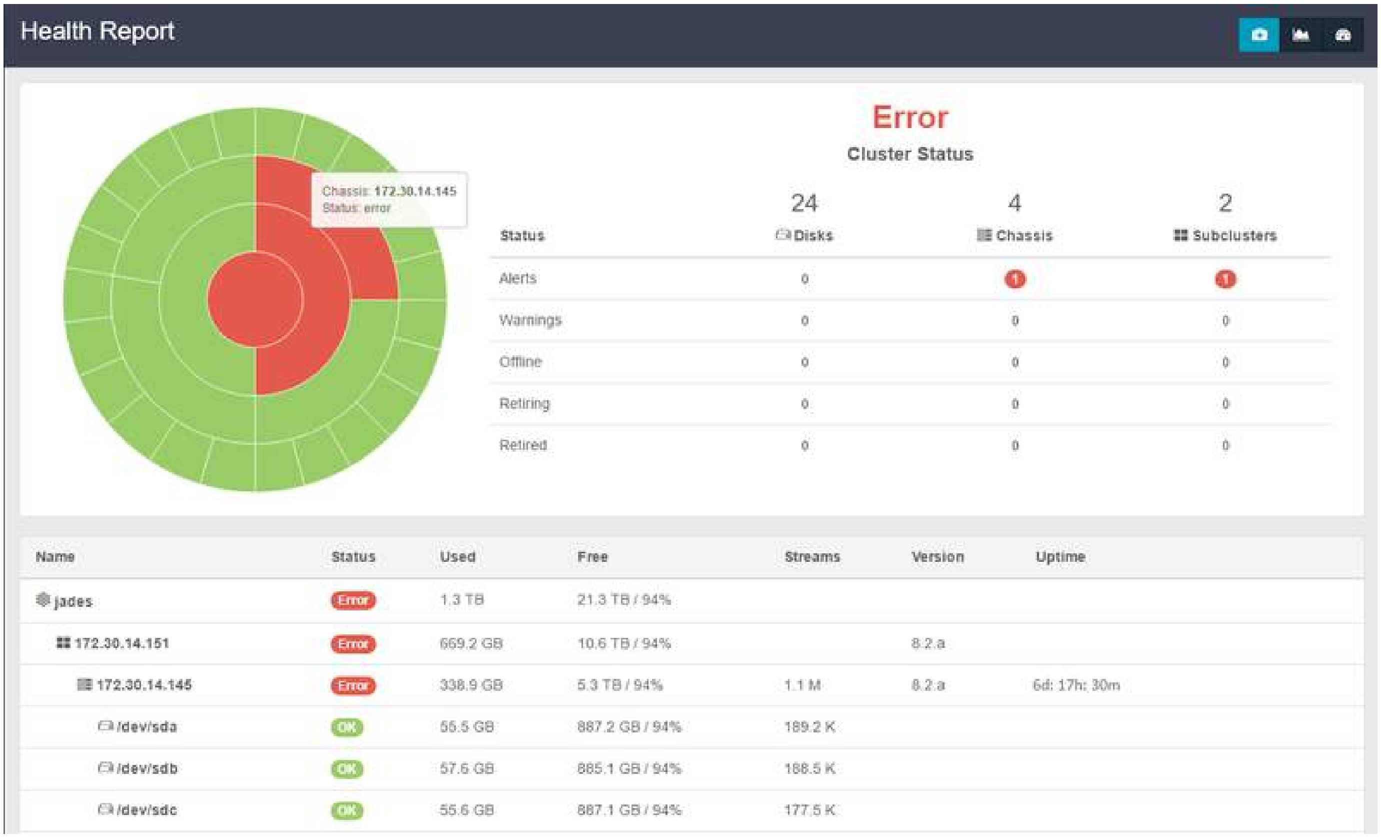
The status of each component is represented by the color of its wedge in the sunburst. Statuses include the following:

OK	The chassis or drive is working and there are no errors.
Alert Warning	The chassis or drive has experienced one or more errors.
Initializing	The short state after a chassis boots when it is reading cluster persisted settings and is not quite ready to accept requests.

Maintenance	The chassis has been shut down or rebooted by an administrator from either SNMP or the UI and should not be considered missing for recovery purposes. By default a chassis can be in a Maintenance state for 3 hours before it transitions to Offline and the cluster starts recovery of its content. Maintenance mode is not initialized when the power is manually cycled on the chassis outside of Swarm (either physically on the hardware or via a remote shutdown mechanism like iDRAC) or if there is a drive error; in both these instances recovery processes will be started for the chassis/drive unless recovery is suspended.
Mounting	The chassis is mounting one or more drives, including formatting the drive if it is new and reading all objects on the volume into the RAM index for faster access.
Offline	The chassis or drive was previously but is no longer present in the cluster.
Retiring	The chassis or drive is in the process of retiring, making sure all its objects are fully protected elsewhere in the cluster and then removing them locally.
Retired	The chassis or drive has completed the retiring process and may be removed from the cluster.
Idle	The chassis or drive is in power-saving mode due during a period of configurable inactivity. (See Configuring Power Management.)

Subcluster and Cluster status is inherited from the chassis or drives contained within.

The data table below the sunburst displays more detailed information about the cluster, including the amount of used and free capacity and how many streams reside on the chassis/drive. Clicking on a subcluster row will take you to the Subcluster page. Clicking on a chassis row will take you to the Chassis Details page unless the chassis status is Maintenance or Offline.



Health Report

Error

Cluster Status

Status	24 Disks	4 Chassis	2 Subclusters
Alerts	0	1	1
Warnings	0	0	0
Offline	0	0	0
Retiring	0	0	0
Retired	0	0	0

Name	Status	Used	Free	Streams	Version	Uptime
jades	Error	1.3 TB	21.3 TB / 94%			
172.30.14.151	Error	669.2 GB	10.6 TB / 94%		8.2.a	
172.30.14.145	Error	338.9 GB	5.3 TB / 94%	1.1 M	8.2.a	6d: 17h: 30m
/dev/sda	OK	55.5 GB	887.2 GB / 94%	189.2 K		
/dev/sdb	OK	57.6 GB	885.1 GB / 94%	188.5 K		
/dev/sdc	OK	55.6 GB	887.1 GB / 94%	177.5 K		

History Reports

The History Reports include several charts that display daily usage and activity in the cluster over a rolling 30-day window. Mouse over the chart to see the exact values for all included data sets on any given day. If there multiple data sets included in the charts, you can toggle on and off the displayed data by clicking on each value in the legend at the bottom of each chart. For instance, to only see Streams on the Storage Contents chart, click the Size label to remove that data from the display.

The History Reports depend on Swarm Historical Metrics to provide historical data for the charts. If Historical Metrics are not available or there is no data available within the last 30 days, the chart will display Data unavailable.

Important

Swarm will generate CRITICAL log messages if Historical Metrics is misconfigured or if connection to the Elasticsearch cluster is lost.

Troubleshooting

Charts will show "Data unavailable" when it cannot obtain Swarm metrics data to display. These are possible causes:

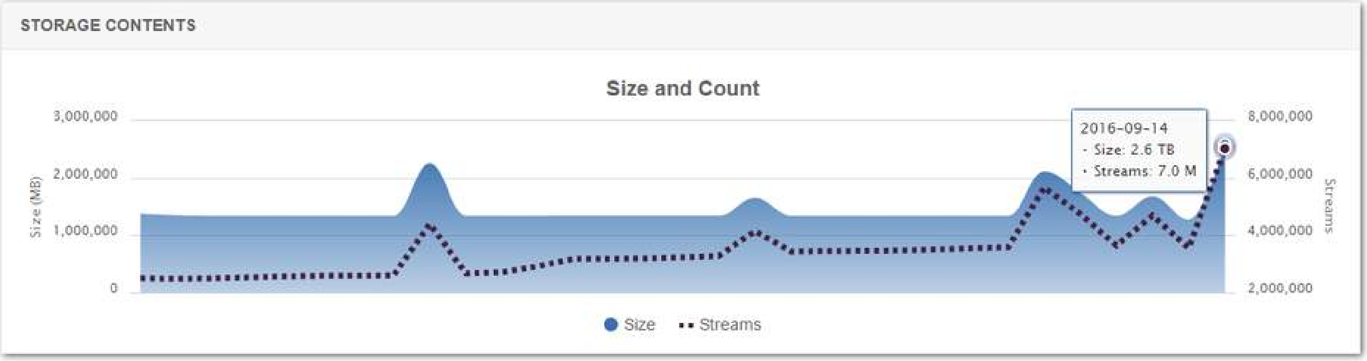
- Metrics data has not yet been recorded
- metrics.targets is not configured (see Installing Swarm Metrics)
- The Metrics Curator is not configured properly or the crond service is not running (see Installing Swarm

Metrics)
- The Gateway Service Proxy cannot contact Elasticsearch (see Service Proxy)
- Elasticsearch is not configured to enable http.cors (see Configuring Elasticsearch)

To help Support resolve your problem, provide the contents of your browser's console (F12 or View > Developer Tools). (UIS-494)

Storage Contents

The Storage Contents chart displays the total amount of used capacity in the cluster over time as well as the total stream count (including replicas and erasure coding segments).



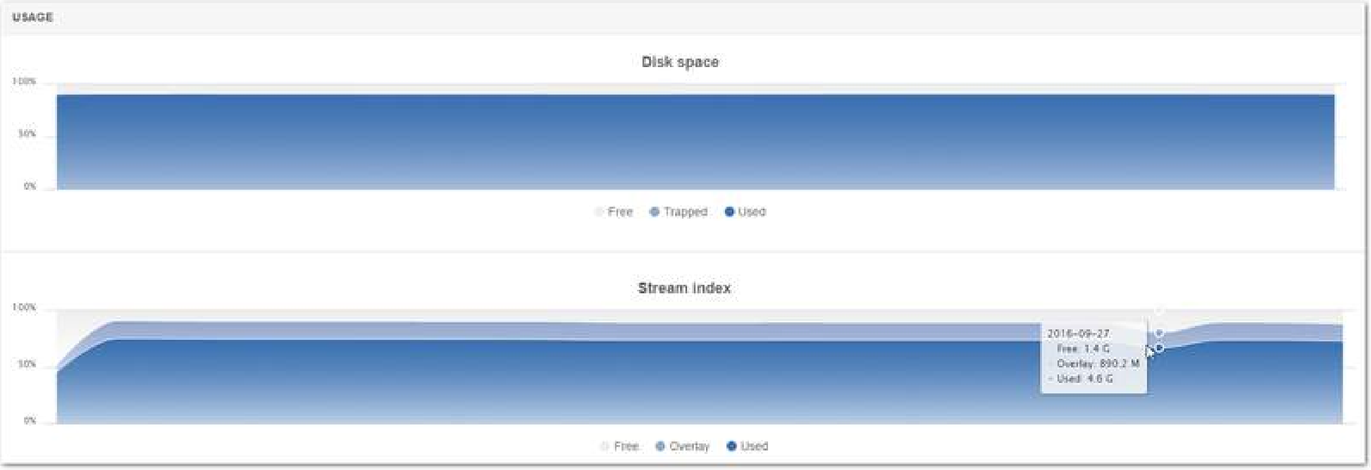
Usage

Note
When you add or remove a large percentage of your drives within a single day, historical usage charts may show artificial bumps in usage for that day.

The Usage charts display percentages of Disk space and Stream index (memory):

Disk space — The amount of free, trapped and used drive space as a percent of the total available over time.

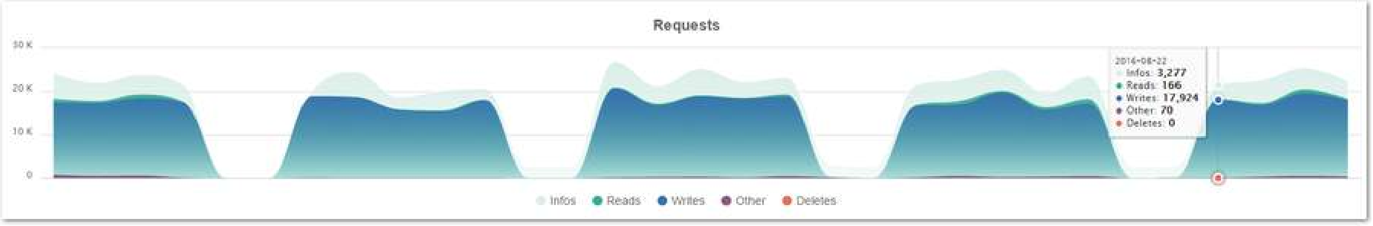
Stream index — The amount of free, overlay, and used RAM index space as a percent of the total available over time.



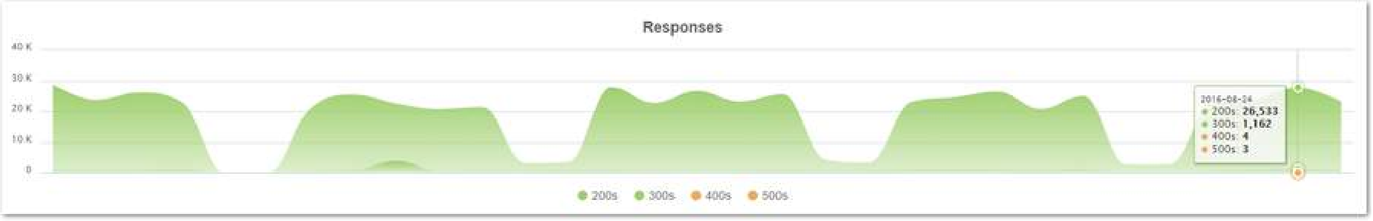
Network Traffic

The Network Traffic graphs display Requests, Responses, and Internal Requests (inter-cluster activity).

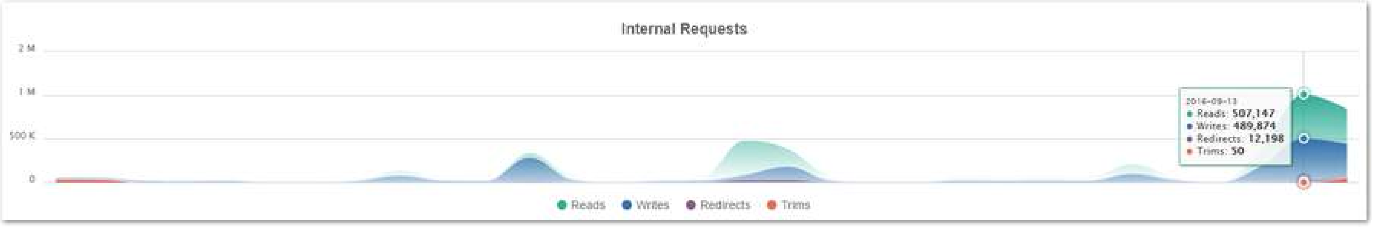
Requests — The count of each SCSP method type in incoming client requests to the cluster over time. SCSP Method types are: Infos, Reads, Writes (sum of writes, updates and appends), Deletes, and Other (sum of metadata updates and Search queries). This information is useful in understanding both when and how your cluster is being used by client applications.



Responses— The count of HTTP response codes returned to clients by the storage cluster over time. This data is helpful in identifying problems in the cluster or client applications, including if there are particular times during which error responses occur.

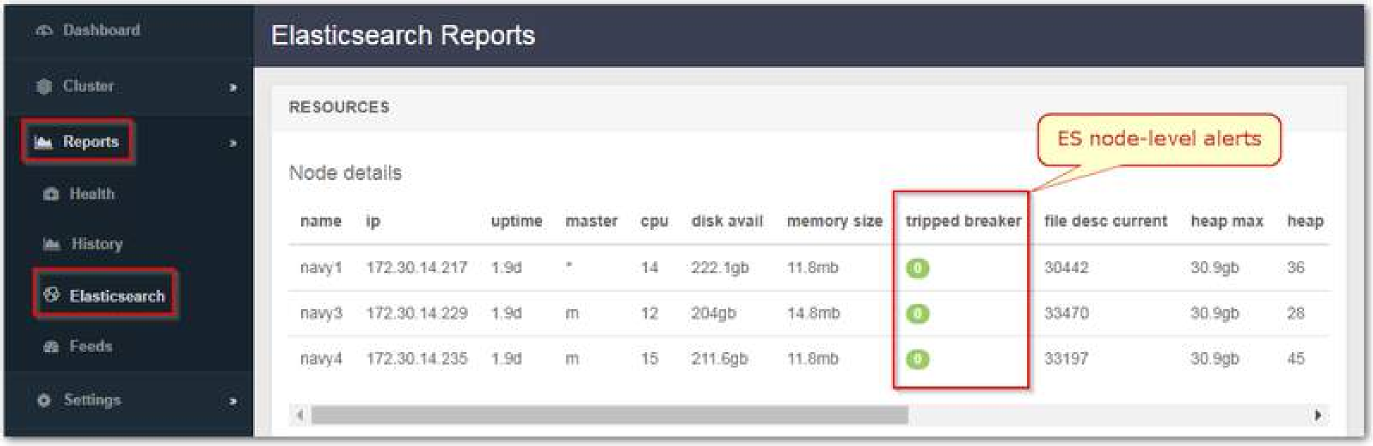


Internal Requests— The count of various internal, cluster initiated activities between nodes in the cluster over time. This information is helpful in understanding how much data movement is happening in the cluster as hardware is added, removed, retired, etc. For instance, spikes in activity within the cluster that don't correlate with client activity are often associated with either a failed drive recovery or an admin requested retire.



Elasticsearch Reports

If the Elasticsearch panel on the Dashboard shows a problem, you can research your ES cluster status on the Elasticsearch Reports page. These reports generate on demand and let you drill into details spanning the ES nodes, thread pools, indices, and shards. (v2.0)



Elasticsearch Reports

RESOURCES

Node details

name	ip	uptime	master	cpu	disk avail	memory size	tripped breaker	file desc current	heap max	heap
navy1	172.30.14.217	1.9d	*	14	222.1gb	11.8mb	0	30442	30.9gb	36
navy3	172.30.14.229	1.9d	m	12	204gb	14.8mb	0	33470	30.9gb	28
navy4	172.30.14.235	1.9d	m	15	211.6gb	11.8mb	0	33197	30.9gb	45

Important

Opening the Elasticsearch Reports page requires generation of a lot of status data; allow time for the page to display.

> For details on the columns that are reported, see the relevant Elasticsearch Reference: version 2.3 or version 5.6.

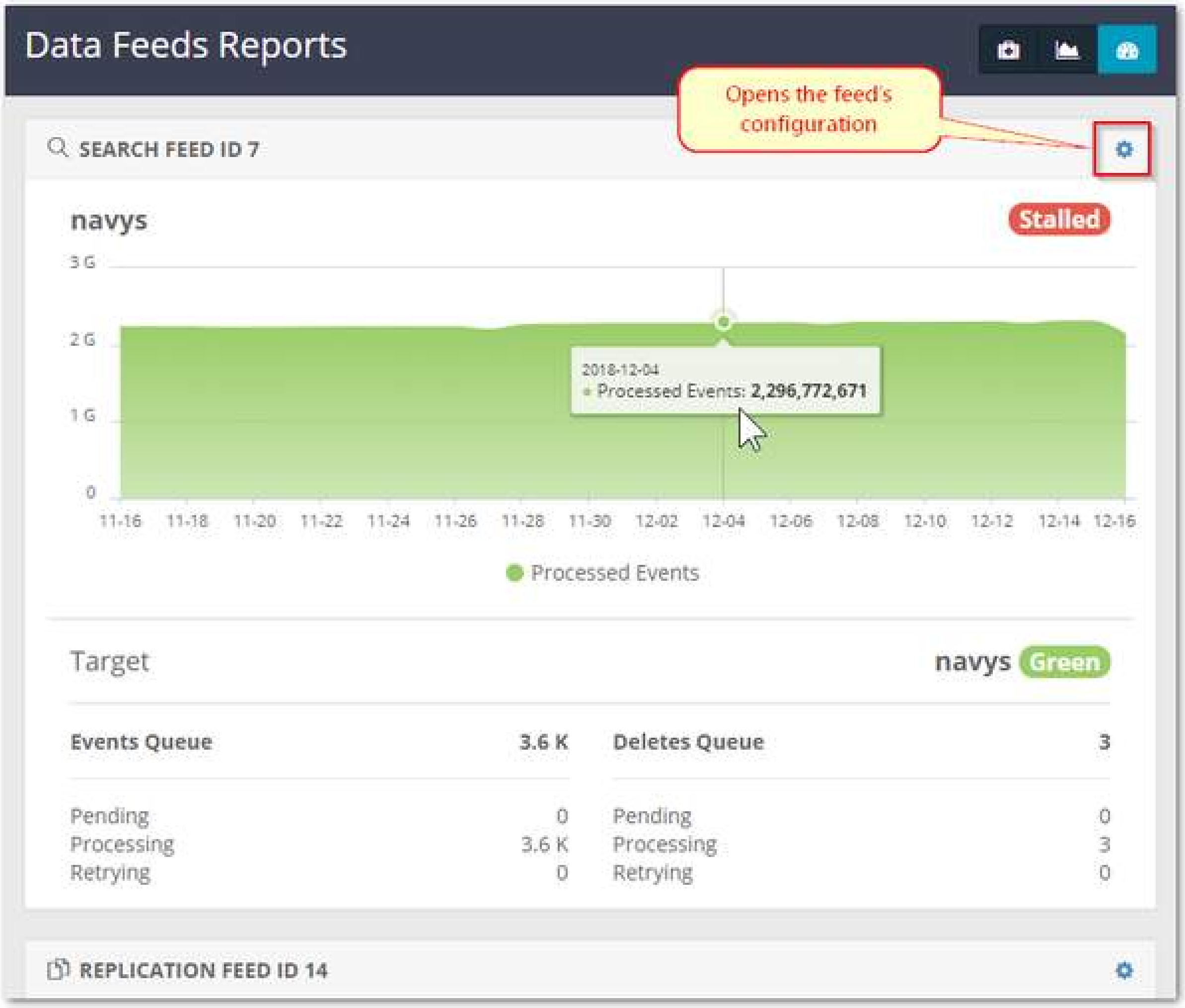
Section	Setting	Notes
RESOURCES Node details	• name • ip • uptime • master • cpu • disk avail • memory size • tripped breaker • file desc current • heap max • heap percent • ram percent • indexing delete total • indexing index total • search query total	Shows the ES cluster topology. For seeing where your nodes live and to check performance stats, focus on these columns: • ip • cpu • tripped breaker Important The tripped breaker field signals trouble. If you see it in a red status, contact Caringo Support. • heap percent • ram percent Other columns are more helpful when looking at larger clusters, such as determining how many master-eligible nodes are available: • master • name

RESOURCES Thread pool details	• name • ip • bulk rejected • flush rejected • force_merge rejected • generic rejected • get rejected • index rejected • refresh rejected • search rejected • warmer rejected	Shows ES cluster-wide thread pool statistics per node. The `rejected` statistics are returned for all thread pools.
INDICES	• index • health • status • docs count • docs deleted • pri • pri store size • rep • store size	Provides low-level information about the segments in the shards of an index. Note Swarm Metrics generates large numbers of indices. docs.count — The number of non-deleted documents that are stored in this segment. These are Lucene documents, so the count includes hidden documents (such as from nested types). docs.deleted — The number of deleted documents that are stored in this segment. The space for these documents will be reclaimed when this segment gets merged
SHARDS	• index • node • ip • docs • prirep • shard • state • store	The detailed view of what nodes contain which shards. It will tell you if it’s a primary or replica, the number of docs, the bytes it takes on disk, and the node where it’s located. prirep — Whether this segment belongs to a primary or replica shard.

Feeds Reports

The Data Feeds Reports show the number of processed events for each configured search or replication feed over time, providing insight into how busy each feed is. Status markers alert you to problems with the feed.

Tip
For quick access to the configuration details for a feed, click the gear icon in the top right of its chart.



Target navys Green

Events Queue	3.6 K	Deletes Queue	3
Pending	0	Pending	0
Processing	3.6 K	Processing	3
Retrying	0	Retrying	0

REPLICATION FEED ID 14

Health Data to Support

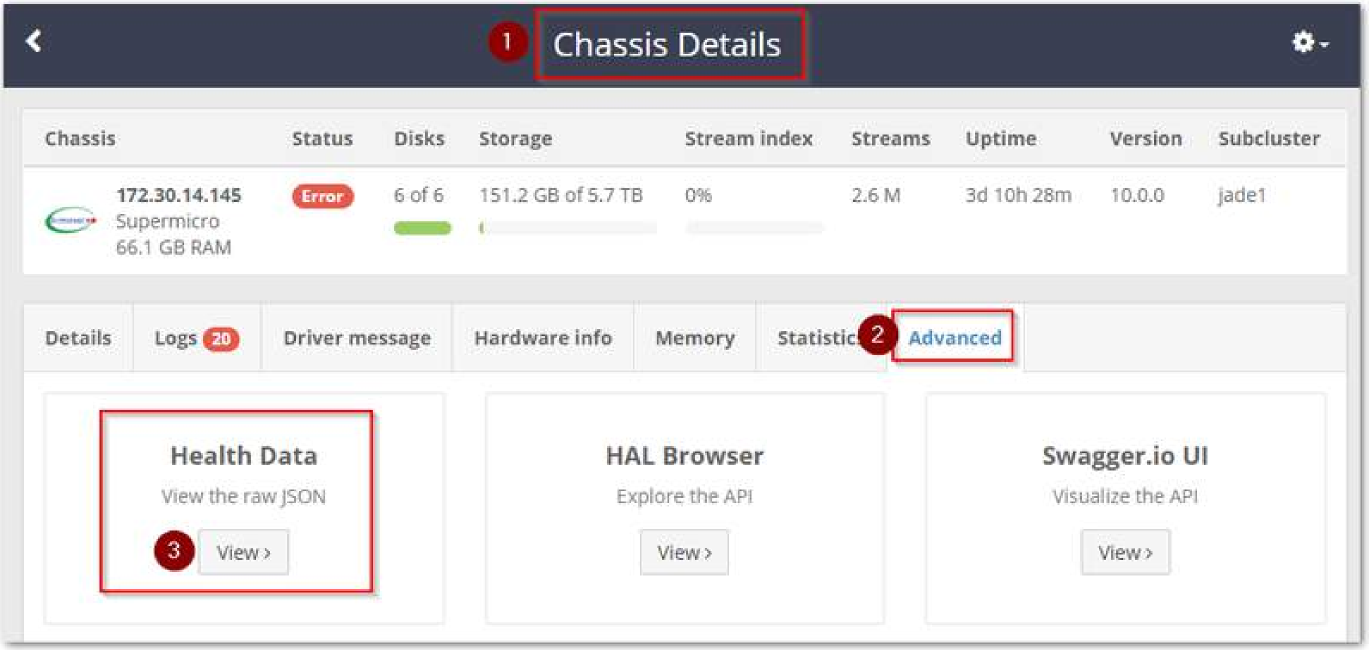
The cluster health report (Health Data) summarizes the information represented in the UI under Reports. This is the health check that your Swarm cluster automatically sends to Caringo Support.

Best practice
Monitoring this health information enables Caringo to provide technical support proactively, so it is best practice to participate in this support service.
However, if you think you need to disable your automatic health reporting, see Support's Health Report article f or details and the FAQ on disabling health reporting.

- Accessing Your Report
- Proxy for Health Reports

Accessing Your Report

To see the health data and what information Caringo receives from your site, you can view the raw report in your browser: Go to Cluster > Hardware, double-click on hardware to open the Chassis Details, then open Advanced, Health Data. (v10.0)



Chassis	Status	Disks	Storage	Stream index	Streams	Uptime	Version	Subcluster
172.30.14.145 Supermicro 66.1 GB RAM	Error	6 of 6	151.2 GB of 5.7 TB	0%	2.6 M	3d 10h 28m	10.0.0	jade1

Deprecated

The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0) To view health reports there, use this URL:

```
<host>:90/health_report
```

Troubleshooting

If you have problems accessing the health report, check the cluster network connectivity to the Internet and verify that the cluster name is populated, there is a cluster settings file, and the `support.uri` parameter has not been disabled. To receive proactive support, you must use the default value "`https://healthreport.caringo.com:443/castor/report`".

Proxy for Health Reports

If you need to specify a proxy specific to health reports, add the following proxy settings as appropriate to your Swarm configuration. Swarm accepts a proxy username and password to allow health reports to be sent via a password-enabled proxy. (v9.2)

Setting	Notes
support.proxyUri	Proxy URI, which may but need not include http(s)://.
support.noProxy	Comma-separated list of domain names or IP addresses for which HTTP/S proxy should not be used. Do not include http(s):// or port numbers. Wildcards are allowed.
support.proxyUsername	Proxy authentication username.
support.proxyPassword	Proxy authentication password.

Legacy Admin Console (port 90)

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- UI Essentials - Legacy Admin Console
- Viewing and Managing the Cluster - Legacy Admin Console
- Managing Chassis and Drives - Legacy Admin Console
- Viewing and Editing Feeds - Legacy Admin Console
- Identifying the Primary Search Feed
- Using Cluster Settings - Legacy Admin Console
- Managing Domains - Legacy Admin Console

UI Essentials - Legacy Admin Console

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Accessing the UI
- Navigating the UI
- Branding the Admin Console

Accessing the UI

To connect to the legacy Admin Console, enter one of the following URLs in your browser's address or location field:

CSN Platform Server

```
http://{CSN·external·IP}:8090/services/storage
```

No Platform Server

```
http://{cluster}:90
```

For example, if the Swarm node IP address is 10.20.30.101, enter:

```
http://10.20.30.101:90
```

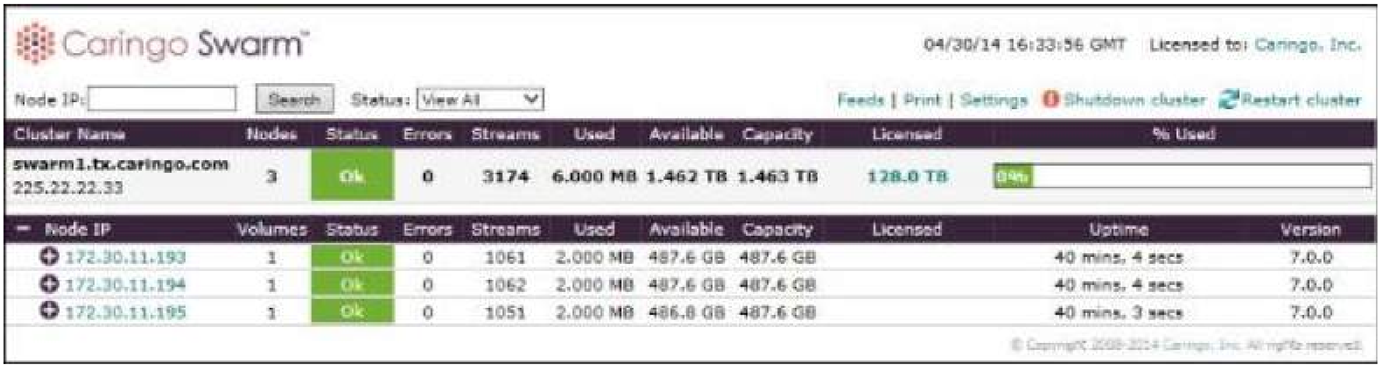
Note
<cluster> in a URL stands for <host>[:<port>], where host is a fully qualified domain name or IP address, plus a port number if other than `80`. If the `Host` header does not match the domain name, override it with the `domain=` argument.

Navigating the UI

The legacy Admin Console displays two separate views of the cluster: cluster-level and node-level status.

Cluster Status Page

When you initially log in to the legacy Admin Console, the Cluster Status page appears.



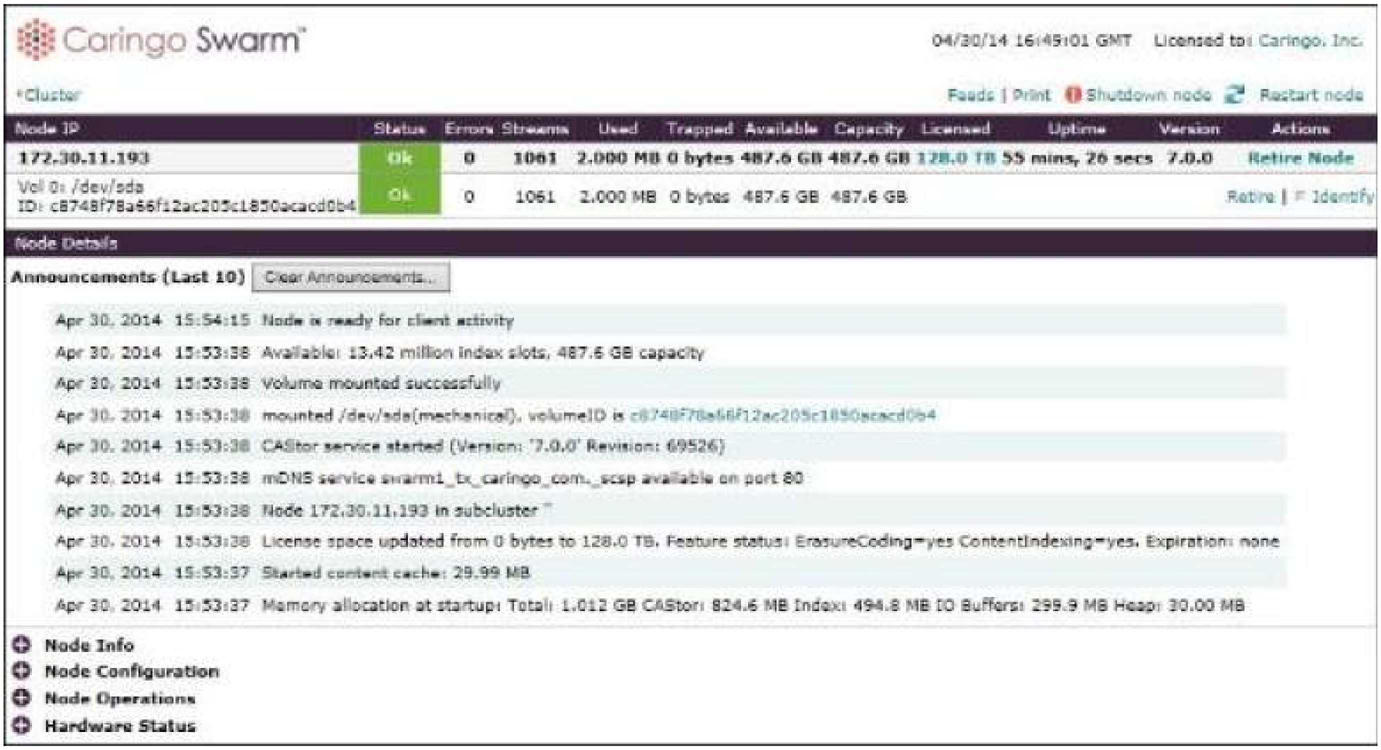
Cluster Name	Nodes	Status	Errors	Streams	Used	Available	Capacity	Licensed	% Used
swarm1.tx.caringo.com 225.22.22.33	3	Ok	0	3174	6.000 MB	1.462 TB	1.463 TB	128.0 TB	0%

− Node IP	Volumes	Status	Errors	Streams	Used	Available	Capacity	Licensed	Uptime	Version
+ 172.30.11.193	1	Ok	0	1061	2.000 MB	487.6 GB	487.6 GB		40 mins, 4 secs	7.0.0
+ 172.30.11.194	1	Ok	0	1062	2.000 MB	487.6 GB	487.6 GB		40 mins, 4 secs	7.0.0
+ 172.30.11.195	1	Ok	0	1051	2.000 MB	486.8 GB	487.6 GB		40 mins, 3 secs	7.0.0

The Cluster Status page lets you view cluster-wide information such as number of nodes, cluster status, number of errors, number of streams (data components that comprise objects), and capacity data. If the cluster is configured with subclusters, you must expand a subcluster node name to display the associated IP addresses and then click an IP address to display node information.

Node Status Page

When you click an IP address in the Node IP column, the Node Status page appears for the selected node, as shown below.



Node IP	Status	Errors	Streams	Used	Trapped	Available	Capacity	Licensed	Uptime	Version	Actions
172.30.11.193	Ok	0	1061	2.000 MB	0 bytes	487.6 GB	487.6 GB	128.0 TB	55 mins, 26 secs	7.0.0	Retire Node
Vol 0: /dev/sda ID: c8748f78a66f12ac205c1850acacd0b4	Ok	0	1061	2.000 MB	0 bytes	487.6 GB	487.6 GB				Retire | Identify

Announcements (Last 10)

```
Apr 30, 2014  15:54:15  Node is ready for client activity
Apr 30, 2014  15:53:38  Available: 13.42 million index slots, 487.6 GB capacity
Apr 30, 2014  15:53:38  Volume mounted successfully
Apr 30, 2014  15:53:38  mounted /dev/sda(mechanical), volumeID is c8748f78a66f12ac205c1850acacd0b4
Apr 30, 2014  15:53:38  CAStor service started (Version: '7.0.0' Revision: 69526)
Apr 30, 2014  15:53:38  mDNS service swarm1_tx_caringo_com._scsp available on port 80
Apr 30, 2014  15:53:38  Node 172.30.11.193 in subcluster ''
Apr 30, 2014  15:53:38  License space updated from 0 bytes to 128.0 TB. Feature status: ErasureCoding=yes ContentIndexing=yes. Expiration: none
Apr 30, 2014  15:53:37  Started content cache: 29.99 MB
Apr 30, 2014  15:53:37  Memory allocation at startup: Total: 1.012 GB CAStor: 824.6 MB Index: 494.8 MB IO Buffers: 299.9 MB Heap: 30.00 MB
```

- Node Info
- Node Configuration
- Node Operations
- Hardware Status

The Node Status page lets you view information specific to your cluster node, such as hardware status, health processor status, uptime, and Swarm software version.

> See Managing Chassis and Drives.

The Node Status page also includes the following sections:

- Node Info. Provides general information about the hardware installed on the node, as well as time server information and current uptime.
- Node Configuration. Provides the cluster and network configuration settings assigned to the node. Use this status information to verify your system configuration quickly, without using SNMP commands.
- Node Operations. Describes the state of the node. If you encounter a problem in your storage cluster, a Swarm Support representative can use the information in this page to help determine if the node is communicating effectively with other nodes and resources in the cluster.
- Hardware Status. Provides status and operational reporting (if available) for various hardware components installed on the node. Use this status information to retrieve node system data, such as the serial number and BIOS version.

Printing the legacy Admin Console

Click Print at the top of the page to display a printer-friendly version of the legacy Admin Console.

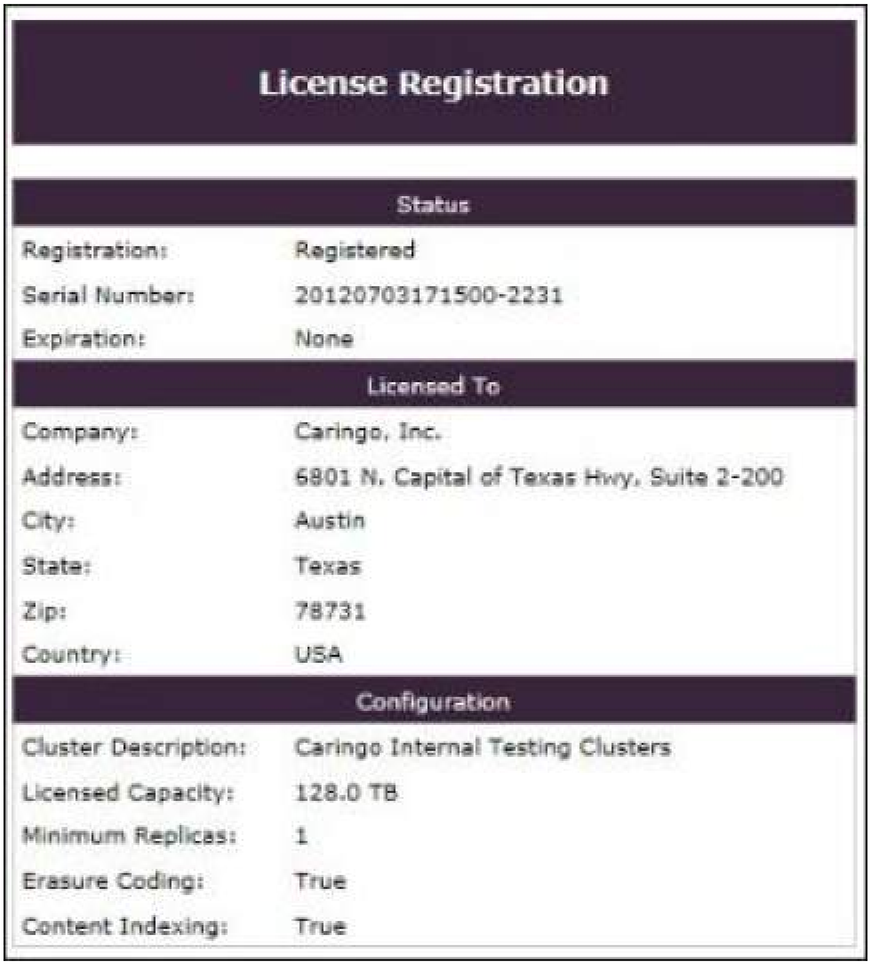
To include a portion of the console in an email, copy the text and paste it to an HTML-formatted email.

If you are not using HTML-formatted email, you can paste a portion of the printed formatted page into an application (such as Microsoft® Excel® or Word®) and then copy it to another location.

> Tip
>
> If you are using Mozilla Firefox®, print the legacy Admin Console in landscape mode to ensure that the image does not extend beyond the right margin.

Viewing License Information

Your cluster license appears when you click the Licensed to link at the top of the legacy Admin Console. This window displays your registration status, contact information, and the configuration settings in the license file.



License Registration

Status	
Registration:	Registered
Serial Number:	20120703171500-2231
Expiration:	None
Licensed To	
Company:	Caringo, Inc.
Address:	6801 N. Capital of Texas Hwy, Suite 2-200
City:	Austin
State:	Texas
Zip:	78731
Country:	USA
Configuration	
Cluster Description:	Caringo Internal Testing Clusters
Licensed Capacity:	128.0 TB
Minimum Replicas:	1
Erasure Coding:	True
Content Indexing:	True

If your cluster license is invalid, Unregistered appears in the Company field and as a watermark in the legacy Admin Console. Contact your Swarm representative to purchase a valid license.

Branding the Admin Console

You can override the baseline style sheets in the legacy Admin Console using a centralized configuration.

The legacy Admin Console ships with a set of default styles. These styles are persisted in these files:

- console.css. Provides a set of baseline styles.
- console_print.css. Provides a small set of overrides for the printed page.
- console_print_preview.css. Provides the styles for the on-screen print view.

These files are loaded at boot time from a USB or centralized configuration `caringo.console` directory. If the files are not available in this location, Swarm uses internal file versions.

The styles defined in the standard files can be overwritten with custom styles on a style-by-style basis. Any styles not overwritten will revert to the baseline styles provided in the default style sheets.

> Tip
> To make customization easier, use a centralized configuration web server.

If a centralized configuration server is not available for your cluster, you can update the console styles by modifying the default styles in console.css, console_print.css, and console_print_preview.css on the USB flash drive for each node.

> Best practice
> Before you modify the default file, create a backup of the file in case you need to revert to the default styles.

Overriding styles on a centralized server

To override the baseline styles using a centralized configuration server:

1. Create new style sheet files, clearly named to distinguish them from the originals. Your new style sheets will define one or more styles from console.css, console_print.css, or console_print_preview.css that you wish to override. At minimum, just paste specific styles from the default style sheets into your new one and then change those style definitions.
2. Install your new styles and place them on any web server that the storage cluster can access.
3. Configure the cluster to reference your new styles with the consoleStyleURL and consoleReportStyleURL node configuration parameters.
   When you override the styles, the consoleReportStyleURL parameter is used for both the console_print.css and console_print_preview.css style sheets.

Viewing and Managing the Cluster - Legacy Admin Console

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Displaying Subcluster Information

This section describes how to manage and maintain your cluster using the legacy Admin Console.

The Cluster Status page appears when you log in to the legacy Admin Console, giving you a comprehensive view of the cluster as a whole and letting you perform cluster-wide actions, such as restarting and shutting down the cluster and modifying the cluster settings.

Authenticating Cluster-wide Actions

Shutting down and restarting the cluster are cluster-wide actions that require authentication. Authentication is also required when changing the cluster settings to:

- Manage domains
- Modify the logging host configuration
- Change the replication multicast value
- Suspend or resume volume recovery
- Set the power-saving mode

To commit your cluster-wide actions,

1. Open the node and/or cluster configuration file with the appropriate user credentials.
2. Modify the security.administrators parameter.

By default, an admin user is predefined in the parameter that lets you authenticate with Basic authentication (your password is sent in clear text from your browser to the legacy Admin Console).

For added security, take the following precautions:

1. Change the admin password immediately.
2. Implement real user names.
3. Encrypt user passwords using Digest authentication.

> See Encrypting Swarm passwords.

> Note
> After 60 seconds in most browsers, your user name and password are no longer valid. Use caution when entering your name and password in Safari and Chrome browsers, where information may not time out after 60 seconds.

Shutting Down or Restarting the Cluster

To shut down or restart all nodes in the cluster,

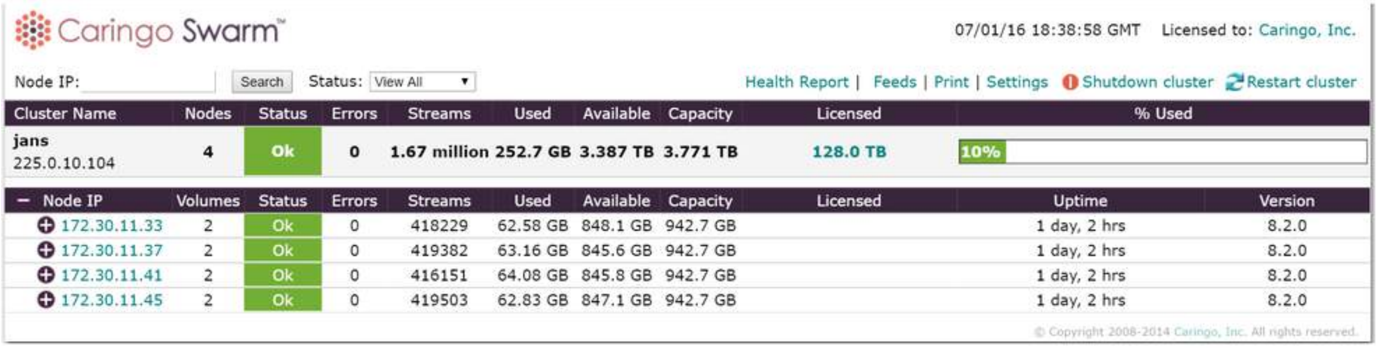
1. Log into the legacy Admin Console with admin credentials.
2. Click Shutdown Cluster or Restart Cluster in the console.
3. When prompted, verify the procedure.

> Note
> Allow several minutes for the nodes to shut down or restart.

Finding Nodes in the Cluster

Listing of nodes

The cluster node list provides a high-level view of the active nodes in the system. Click the maximize button next to each IP address in the Node IP column to view the storage volumes within each cluster node.


Caringo Swarm™
07/01/16 18:38:58 GMT Licensed to: Caringo, Inc.
Node IP:
Search
Status: View All
Health Report | Feeds | Print | Settings | Shutdown cluster | Restart cluster
© Copyright 2008-2014 Caringo, Inc. All rights reserved.

Cluster Name	Nodes	Status	Errors	Streams	Used	Available	Capacity	Licensed	% Used
jans 225.0.10.104	4	Ok	0	1.67 million	252.7 GB	3.387 TB	3.771 TB	128.0 TB	10%

− Node IP	Volumes	Status	Errors	Streams	Used	Available	Capacity	Licensed	Uptime	Version
⊕ 172.30.11.33	2	Ok	0	418229	62.58 GB	848.1 GB	942.7 GB		1 day, 2 hrs	8.2.0
⊕ 172.30.11.37	2	Ok	0	419382	63.16 GB	845.6 GB	942.7 GB		1 day, 2 hrs	8.2.0
⊕ 172.30.11.41	2	Ok	0	416151	64.08 GB	845.8 GB	942.7 GB		1 day, 2 hrs	8.2.0
⊕ 172.30.11.45	2	Ok	0	419503	62.83 GB	847.1 GB	942.7 GB		1 day, 2 hrs	8.2.0

The status information is transmitted periodically to the legacy Admin Console, requiring up to two minutes before the node data in the Cluster Status window is updated. Because of the status propagation delay, the data for each node may vary in comparison. For best results, remain connected to the same node to avoid confusion.

The volume labels next to each Swarm node are listed in arbitrary order. While the legacy Admin Console labels do not correspond to physical drive slots in node chassis, the volume names match the physical drives in the machine chassis. If the cluster is configured to use subclusters, you must expand each subcluster name to display the corresponding volume information.

> Prior versions
>
> Nodes running legacy software versions (up to version 3.0) in a mixed version configuration may not display all data in the legacy Admin Console, such as object counts.

If you have a large cluster, you can search for nodes by IP address and by status.

Finding nodes by IP

For large clusters with multiple nodes, you can search for a node using the Node IP search field in the legacy Admin Console.

To locate the targeted node, enter the node IP address in the field and click Search .

Finding nodes by Status

To display nodes or volumes with a specific status, select a Status from the drop-down menu.

> Note
>
> The overall cluster status is a rollup of the statuses from cluster nodes.

Statuses include:

- Ok: The node is working and there are no errors.
- Alert, Warning: The node or volume has experienced one or more errors. Click the IP Address link to drill down to the node and view the related error.
- Initializing: The short state after a node boots when it is reading cluster persisted settings and is not quite ready

to accept requests.

- Maintenance: The node has been shut down or rebooted by an administrator from either SNMP or the legacy Admin Console and should not be considered missing for recovery purposes. By default a node can be in a Maintenance state for 3 hours before it transitions to Offline and the cluster starts recovery of its content. Maintenance mode is not initialized when the power is manually cycled on the node outside of Swarm (either physically on the hardware or via a remote shutdown mechanism like iDRAC) or if there is a disk error; in both these instances recovery processes will be started for the node unless recovery is suspended.
- Mounting: The node is mounting one or more volumes, including formatting the disk if it is new and reading all objects on the volume into the RAM index for faster access.
- Offline: The node or volume was previously but is no longer present in the cluster.
- Retiring: The node or volume is in the process of retiring, making sure all its objects are fully protected elsewhere in the cluster and then removing them locally.
- Retired: The node or volume has completed the retiring process and may be removed from the cluster.
- Idle: The nodes or volumes are in power-saving mode due during a period of configurable inactivity. (See Configuring Power Management.)

Only matching results will appear on the console when you select a value in the drop-down menu. When you are finished looking at the searched node(s), select View All to redisplay all nodes in the cluster.

Percent Used Indicator

The % Used indicator provides a helpful computation of cluster availability and licensed and total physical space for monitoring purposes. Space used is calculated against the lesser of the total physical space or the licensed space.

For example, in a cluster with 4 TB of physical space but only 2 TB of licensed space where 1.5 TB of space is used, the console would report 75% Space Used.

The indicators include color highlighting, as described below.

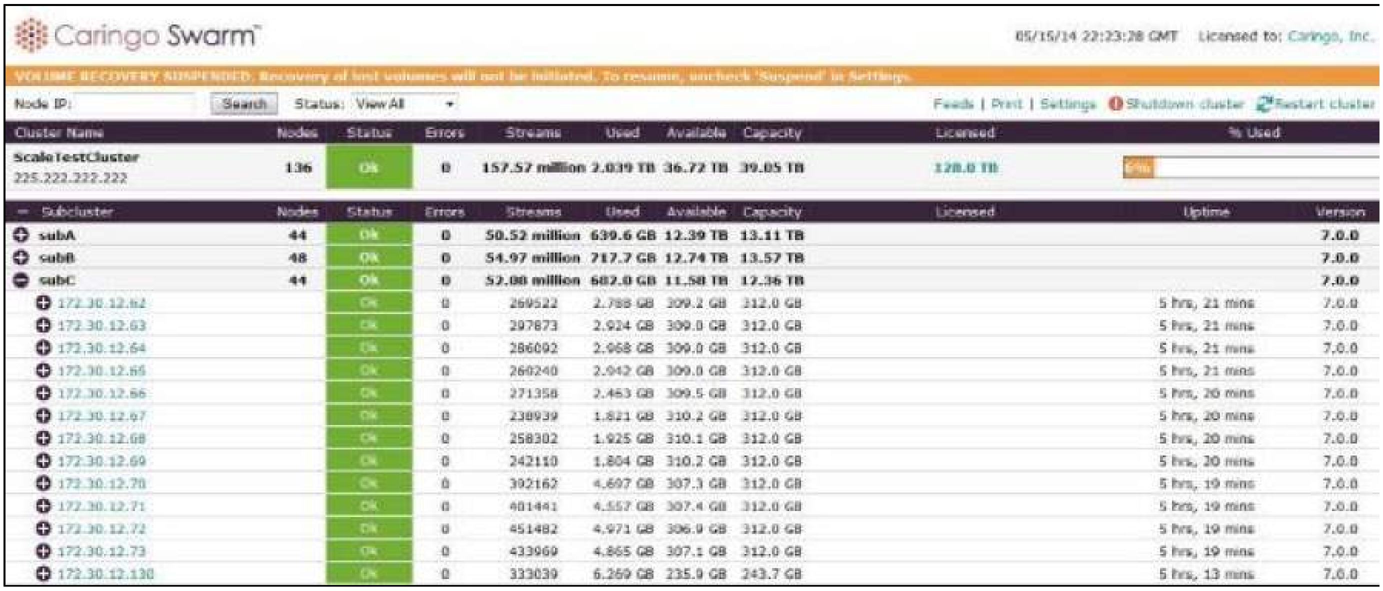
Logical Threshold	Color *	Description	Default Threshold Value
OK	Green	Used space is less than the console.spaceWarnLevel configurable threshold.	At or above 75%
Warning	Yellow	Used space is less than the console.spaceErrorLevel and more than the spaceWarnLevel configurable thresholds.	Above 75% but at or below 90%
Error	Red	Used space is greater than or equal to the console.spaceErrorLevel configurable threshold.	Above 90%

* You can modify these default colors using custom style sheets.

Displaying Subcluster Information

If your cluster contains subclusters, the Node List will be grouped first by subcluster name and then by node IP address. (If no subcluster name is specified in the node or cluster configuration file, the subcluster name is an IP address.) The first row of each subcluster includes a roll up of the status for the nodes in the subcluster.

Example of two subclusters expanded to show member nodes:



05/15/14 22:23:28 GMT Licensed to: Caringo, Inc.

VOLUME RECOVERY SUSPENDED. Recovery of lost volumes will not be initiated. To resume, uncheck 'Suspend' in Settings.

Cluster Name	Nodes	Status	Errors	Streams	Used	Available	Capacity	Licensed	% Used
ScaleTestCluster 225.222.222.222	136	Ok	0	157.57 million	2.039 TB	36.72 TB	39.05 TB	128.0 TB	[illegible]

− Subcluster	Nodes	Status	Errors	Streams	Used	Available	Capacity	Licensed	Uptime	Version
subA	44	Ok	0	50.52 million	639.6 GB	12.39 TB	13.11 TB			7.0.0
subB	48	Ok	0	54.97 million	717.7 GB	12.74 TB	13.57 TB			7.0.0
subC	44	Ok	0	52.08 million	682.0 GB	11.58 TB	12.36 TB			7.0.0
172.30.12.62		Ok	0	269522	2.788 GB	309.2 GB	312.0 GB		5 hrs, 21 mins	7.0.0
172.30.12.63		Ok	0	297873	2.924 GB	309.0 GB	312.0 GB		5 hrs, 21 mins	7.0.0
172.30.12.64		Ok	0	286092	2.968 GB	309.0 GB	312.0 GB		5 hrs, 21 mins	7.0.0
172.30.12.65		Ok	0	260240	2.942 GB	309.0 GB	312.0 GB		5 hrs, 21 mins	7.0.0
172.30.12.66		Ok	0	271358	2.463 GB	309.5 GB	312.0 GB		5 hrs, 20 mins	7.0.0
172.30.12.67		Ok	0	238939	1.821 GB	310.2 GB	312.0 GB		5 hrs, 20 mins	7.0.0
172.30.12.68		Ok	0	258302	1.925 GB	310.1 GB	312.0 GB		5 hrs, 20 mins	7.0.0
172.30.12.69		Ok	0	242110	1.804 GB	310.2 GB	312.0 GB		5 hrs, 20 mins	7.0.0
172.30.12.70		Ok	0	392162	4.697 GB	307.3 GB	312.0 GB		5 hrs, 19 mins	7.0.0
172.30.12.71		Ok	0	401441	4.557 GB	307.4 GB	312.0 GB		5 hrs, 19 mins	7.0.0
172.30.12.72		Ok	0	451482	4.971 GB	306.9 GB	312.0 GB		5 hrs, 19 mins	7.0.0
172.30.12.73		Ok	0	433969	4.865 GB	307.1 GB	312.0 GB		5 hrs, 19 mins	7.0.0
172.30.12.130		Ok	0	333039	6.269 GB	235.9 GB	243.7 GB		5 hrs, 13 mins	7.0.0

The status information is transmitted periodically to the legacy Admin Console, requiring up to two minutes before the node data in the Cluster Status page is updated. Because of the status propagation delay, the data for each node may vary in comparison.

> Tip
> For best results, remain connected to the same node to avoid confusion.

Managing Chassis and Drives - Legacy Admin Console

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

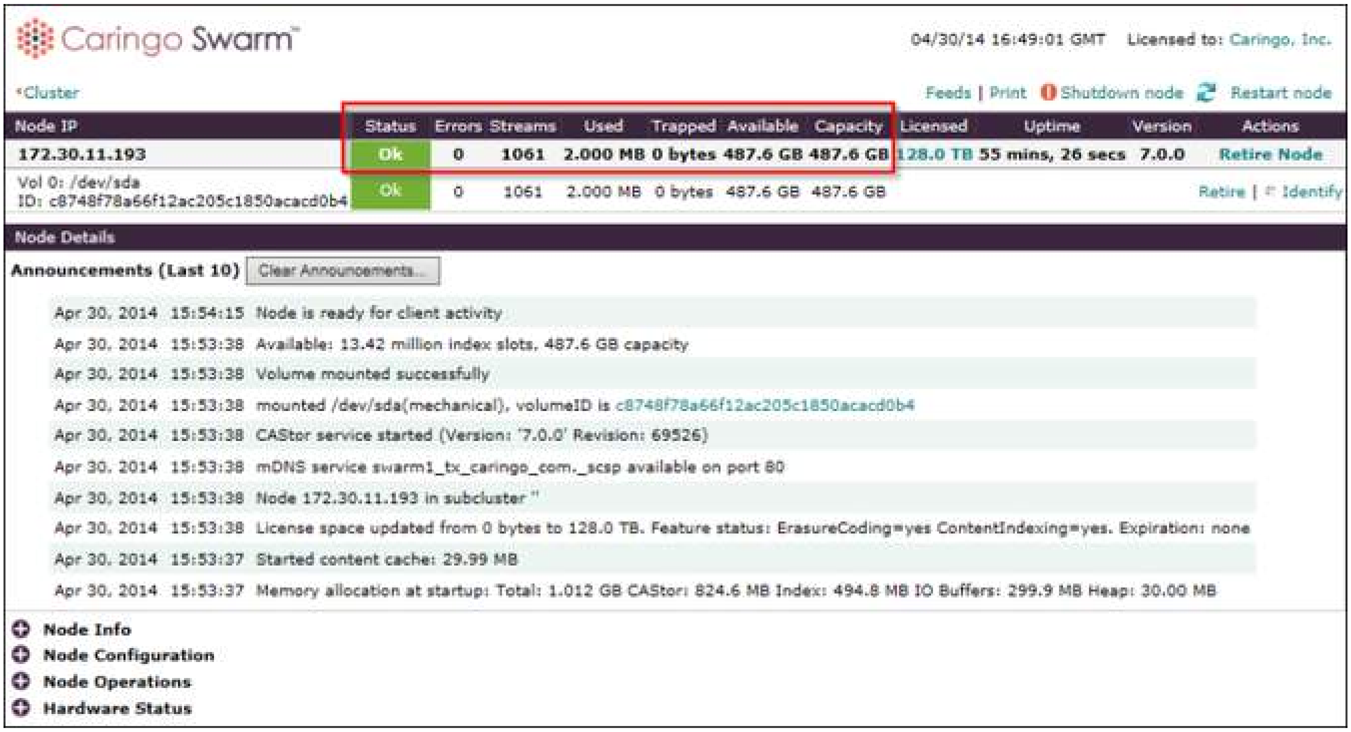
- Viewing the Node Status Page
- Shutting Down or Restarting a Node
- Identifying a Drive
- Retiring a Drive
- Errors and Announcements
- Node Status Reporting

Viewing the Node Status Page

To view the status of a node, click its IP address on the left side of the Swarm Admin Console. If the cluster is configured to use subclusters, expand a subcluster node name to display IP addresses and then click an IP address to display the node information.

> To find a particular node, see Searching for Nodes by IP Address.

The top row of the Node Status page provides summary information about the node and its associated volumes, such as up-time and storage usage statistics:



Node IP	Status	Errors	Streams	Used	Trapped	Available	Capacity	Licensed	Uptime	Version	Actions
172.30.11.193	Ok	0	1061	2.000 MB	0 bytes	487.6 GB	487.6 GB	128.0 TB	55 mins, 26 secs	7.0.0	Retire Node
Vol 0: /dev/sda ID: c8748f78a66f12ac205c1850acacd0b4	Ok	0	1061	2.000 MB	0 bytes	487.6 GB	487.6 GB				Retire | Identify

Announcements (Last 10)

```
Apr 30, 2014  15:54:15  Node is ready for client activity
Apr 30, 2014  15:53:38  Available: 13.42 million index slots, 487.6 GB capacity
Apr 30, 2014  15:53:38  Volume mounted successfully
Apr 30, 2014  15:53:38  mounted /dev/sda(mechanical), volumeID is c8748f78a66f12ac205c1850acacd0b4
Apr 30, 2014  15:53:38  CAStor service started (Version: '7.0.0' Revision: 69526)
Apr 30, 2014  15:53:38  mDNS service swarm1_tx_caringo_com._scsp available on port 80
Apr 30, 2014  15:53:38  Node 172.30.11.193 in subcluster ''
Apr 30, 2014  15:53:38  License space updated from 0 bytes to 128.0 TB. Feature status: ErasureCoding=yes ContentIndexing=yes. Expiration: none
Apr 30, 2014  15:53:37  Started content cache: 29.99 MB
Apr 30, 2014  15:53:37  Memory allocation at startup: Total: 1.012 GB CAStor: 824.6 MB Index: 494.8 MB IO Buffers: 299.9 MB Heap: 30.00 MB
```

- Streams: Counts the total number of managed data components (such as replicas and segments), not logical objects (such as video files).
- Trapped: Calculates the space that is pending reclamation by the Swarm defragmentation process. This process is controlled by several Swarm parameters (see the Settings Reference).

> Note
> The node status page automatically refreshes every 30 seconds.

Shutting Down or Restarting a Node

To shut down or restart a node,

- Click Shutdown Node or Restart Node in the Swarm Admin Console.

A node that is shutdown or rebooted by an Administrator will appear with a Maintenance state on other nodes in the cluster.

> See Searching for Nodes by Status.

> Important
> If you use a multi-server configuration (see Implementing a Multi-Server Chassis), be aware that shutdown or restart impacts all nodes in the multi-server chassis because all nodes in that chassis share system resources.

Identifying a Drive

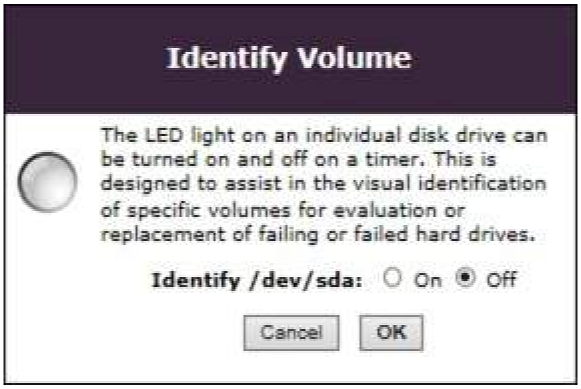
You can identify one or all volumes on a node using the links on the right side of the Swarm Admin Console under Rest art Node.

The Identify function allows you to select a particular volume and enable the corresponding LED drive light, which can be helpful in identifying a failed or failing drive. Simply select the targeted volume and the amount of time that the light will be enabled.

On the Node Status page, an Identify light displays next to the targeted volume for easy identification.

> See Drive Identification Plugin for how to create a plug-in API to enable the drive light.

If a hardware-specific API is not used, Swarm will revert to a default process to flash the light.



Identify Volume

The LED light on an individual disk drive can be turned on and off on a timer. This is designed to assist in the visual identification of specific volumes for evaluation or replacement of failing or failed hard drives.

Identify /dev/sda: ○ On ● Off

Retiring a Drive

You can retire one or all volumes on a node using the links on the right side of the Swarm Admin Console under Restart Node.

On occasion, you may need to replace Swarm volumes for regular maintenance or to upgrade the cluster nodes with higher capacity drives. If multiple volumes need to be replaced across multiple Swarm nodes, the volumes should be retired one at a time. When you initiate a retire, you choose if you would like a minimally disruptive retire that is limited to just the volume(s) being retired, or an accelerated retire that uses all nodes in the cluster to replicate objects on the retiring volume(s) as quickly as possible. Note that the cluster-wide retire may impact performance as it does put additional load on the cluster.

Clicking Retire Node retires all volumes on the node, at the same time. Clicking Retire next to a volume retires only the selected volume. A volume is also retired automatically if a configurable number of errors occur.

> See Retiring Volumes.

Before you retire a node or volume, make sure that the cluster:

- Has enough capacity for the objects on the retiring node to replicate elsewhere.
- Has enough unique nodes to replicate the objects with only one replica on any given node.

> Note
>
> Retire succeeds only if objects can be replicated elsewhere in the cluster. As a result, the Retire action will not remove an object until it can guarantee that at least two replicas exist in the cluster or the existing number of replicas matches the policy.replicas min parameter value.

A retiring node or volume accepts no new or updated objects. Retiring a node or volume means all of its objects, including replicas, are copied to other nodes in the cluster.

On the Swarm Admin Console's Node Status page, the Node Operations section includes a Retire Rate that tracks the number of objects per hour that were removed from a retiring volume. The SNMP MIB includes this same value in the retireRatePerHour MIB entry. If no volumes on the node are retiring, the value is 0.

After all objects are copied, the node or volume's state changes to Retired and Swarm no longer uses the node or volume. At this point, you should remove and repair the volume or discard it.

Errors and Announcements

The last 10 errors and announcements appear on the Node Status page. If there are no errors or announcements, the page is blank. The error count in the node summary grid corresponds to the list of errors in the error section.

> Tip
>
> You can control how long uncleared error messages continue to appear in the error table by configuring the Swarm setting console.messageExpirationSeconds, which defaults to two weeks.

Messages display in the node status area if you remove or insert a drive into a running node. This feature, referred to as hot plugging (adding a new drive) or hot swapping (replacing a failed drive), lets you remove failed drives for analysis or to add storage capacity to a node at any time.

For example, when you add a volume, the following message appears:

```
mounted /dev/sdb, volumeID is 561479FB832DCC526B1D7EDCD06B83E1
```

When you remove a volume, the following message appears:

```
removed /dev/sdb, volumeID was 561479FB832DCC526B1D7EDCD06B83E1
```

> Note
>
> These messages appear at the announcement level. Additional debug level messages appear in the syslog.

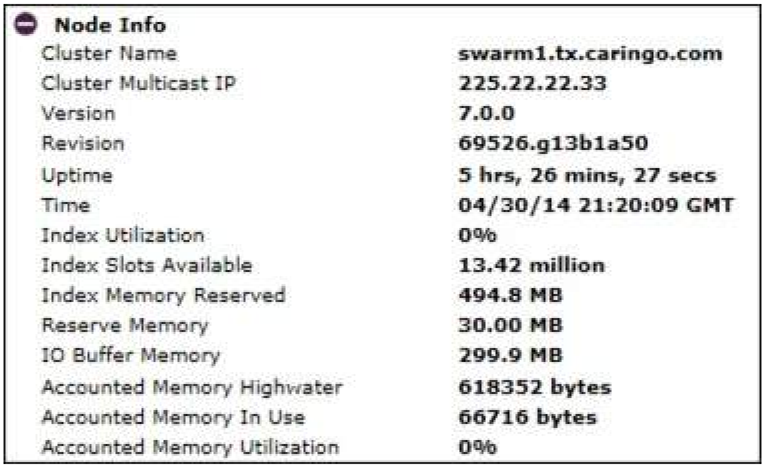
Node Status Reporting

You can troubleshoot node errors and announcements by viewing the reporting sections in the Node Status page. You can access these sections at the bottom of the Node Status page. The information in each section can be helpful when working with Swarm Support to resolve an issue.

Node Info section

The Node Info status section contains general information about the hardware installed on the node, as well as time server information and current uptime. Use this status information to determine if your node requires additional hardware resources.

For example, if the Index Utilization and Buffer Utilization values rise to 80% or more, the Swarm Admin Console generates an alert that indicates the node may require additional RAM to maintain cluster performance. Additionally, if the Time value does not match the same value in the remaining cluster nodes, the node may not be communicating properly with an NTP server.



⊖ Node Info	
Cluster Name	swarm1.tx.caringo.com
Cluster Multicast IP	225.22.22.33
Version	7.0.0
Revision	69526.g13b1a50
Uptime	5 hrs, 26 mins, 27 secs
Time	04/30/14 21:20:09 GMT
Index Utilization	0%
Index Slots Available	13.42 million
Index Memory Reserved	494.8 MB
Reserve Memory	30.00 MB
IO Buffer Memory	299.9 MB
Accounted Memory Highwater	618352 bytes
Accounted Memory In Use	66716 bytes
Accounted Memory Utilization	0%

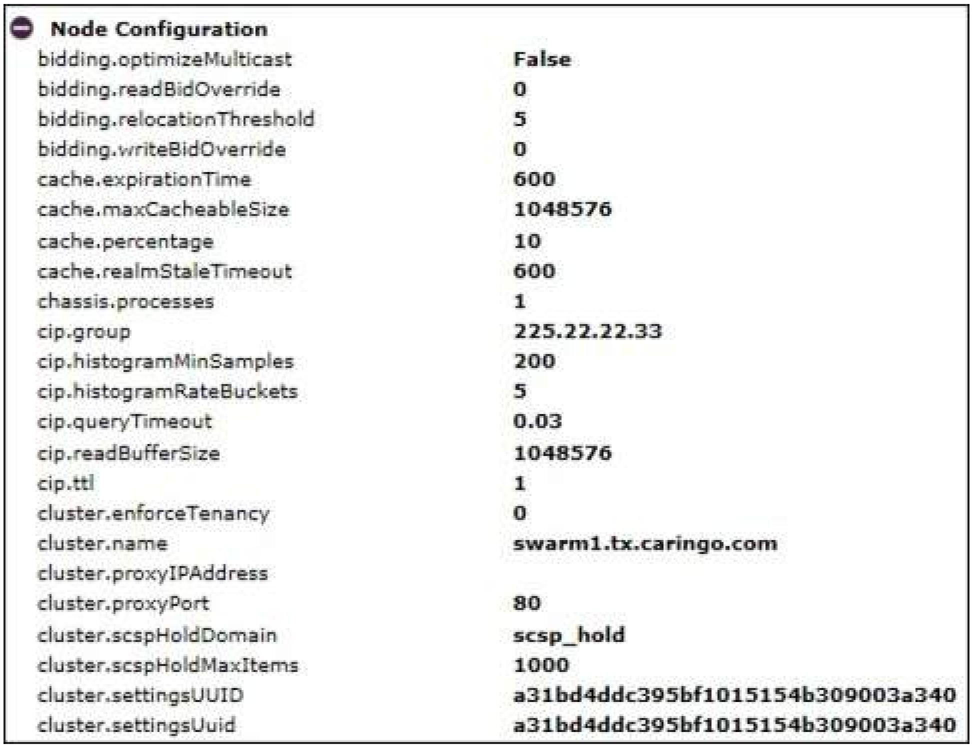
Additional Node Info reports

Scroll to the bottom of the Node Info section to access these links to additional reports:

- SNMP Repository (the SNMP repository dump)
- Object Counts (the Python classes in use)
- Uncollectable Garbage
- HTML Templates
- Loggers... (the settings window for changing the logging levels)
- Dmesg dump (the last 1000 messages logged by the Linux kernel reading buffer, for diagnosing a Swarm issue when a system panic or error occurs)
- Hwinfo dump (the Linux hardware detection tool output)

Node Configuration section

The Node Configuration status section contains the cluster and network configuration settings assigned to the node. Use this status information to quickly verify your system configuration without using SNMP commands.



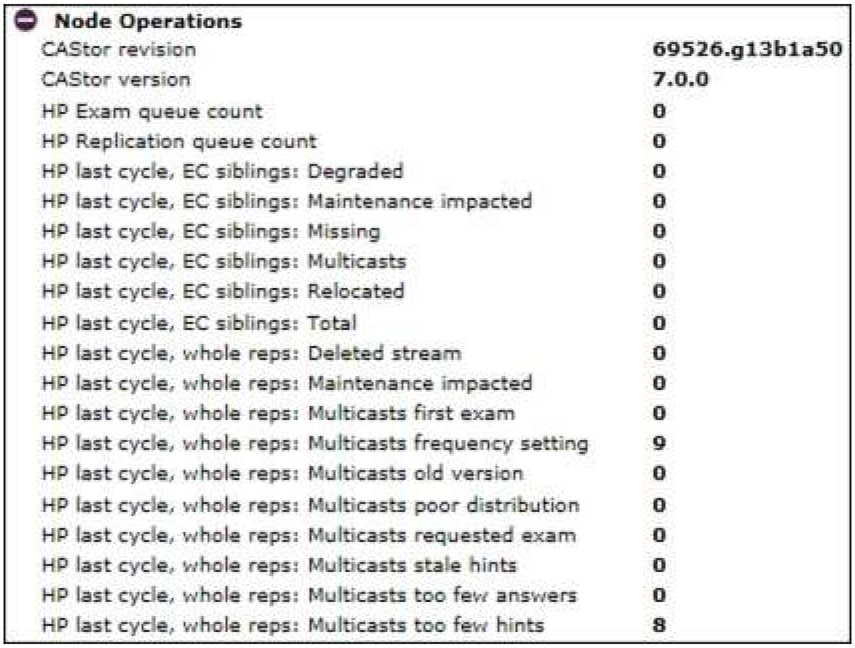
Node Configuration

bidding.optimizeMulticast	False
bidding.readBidOverride	0
bidding.relocationThreshold	5
bidding.writeBidOverride	0
cache.expirationTime	600
cache.maxCacheableSize	1048576
cache.percentage	10
cache.realmStaleTimeout	600
chassis.processes	1
cip.group	225.22.22.33
cip.histogramMinSamples	200
cip.histogramRateBuckets	5
cip.queryTimeout	0.03
cip.readBufferSize	1048576
cip.ttl	1
cluster.enforceTenancy	0
cluster.name	swarm1.tx.caringo.com
cluster.proxyIPAddress	
cluster.proxyPort	80
cluster.scspHoldDomain	scsp_hold
cluster.scspHoldMaxItems	1000
cluster.settingsUUID	a31bd4ddc395bf1015154b309003a340
cluster.settingsUuid	a31bd4ddc395bf1015154b309003a340

Node Operations section

The Node Operations status section describes the state of the node. If you encounter a problem in your storage cluster, a Swarm Support representative can use the information in this page to help you determine if the node is communicating effectively with other nodes and resources in the cluster.

For example, some cluster features (such as the Capacity column value in the Swarm Admin Console) will not update until the HP cycles are completed separately on each node. The HP Cycle time parameter increases exponentially as the number of objects increase on the node. Additionally, if the SCSP Last read bid and SCSP Last write bid parameters are high, the node may not be servicing new requests.



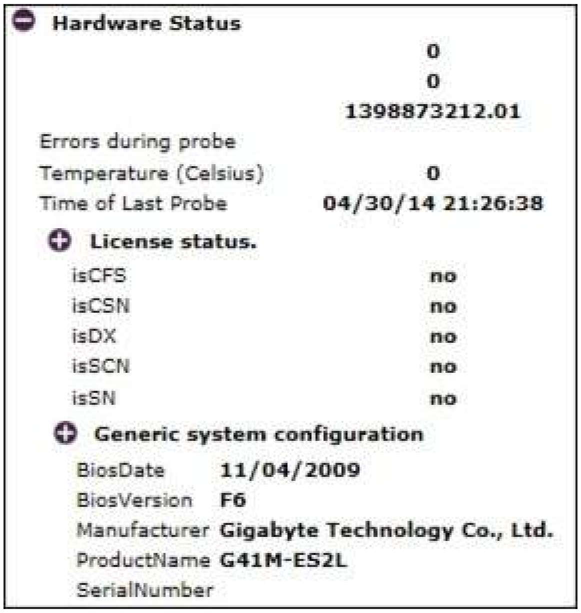
⊖ **Node Operations**

CAStor revision	69526.g13b1a50
CAStor version	7.0.0
HP Exam queue count	0
HP Replication queue count	0
HP last cycle, EC siblings: Degraded	0
HP last cycle, EC siblings: Maintenance impacted	0
HP last cycle, EC siblings: Missing	0
HP last cycle, EC siblings: Multicasts	0
HP last cycle, EC siblings: Relocated	0
HP last cycle, EC siblings: Total	0
HP last cycle, whole reps: Deleted stream	0
HP last cycle, whole reps: Maintenance impacted	0
HP last cycle, whole reps: Multicasts first exam	0
HP last cycle, whole reps: Multicasts frequency setting	9
HP last cycle, whole reps: Multicasts old version	0
HP last cycle, whole reps: Multicasts poor distribution	0
HP last cycle, whole reps: Multicasts requested exam	0
HP last cycle, whole reps: Multicasts stale hints	0
HP last cycle, whole reps: Multicasts too few answers	0
HP last cycle, whole reps: Multicasts too few hints	8

Hardware Status section

The Hardware Status section contains status and operational reporting (if available) for various hardware components installed on the node. Use this status information to retrieve node system data, such as the serial number and BIOS version.

Hardware status reporting is dependent on hardware that supports and populates IPMI sensors, SMART status, and, in some cases, manufacturer-specific components such as SAS. Depending on your hardware, not all status fields are populated. The hardware status values are independently scanned and populated for each node, allowing variations in supported utilities on a node-by-node basis.



⊖ **Hardware Status**

	0
	0
	1398873212.01
Errors during probe	
Temperature (Celsius)	0
Time of Last Probe	04/30/14 21:26:38

⊕ **License status.**

isCFS	no
isCSN	no
isDX	no
isSCN	no
isSN	no

⊕ **Generic system configuration**

BiosDate	11/04/2009
BiosVersion	F6
Manufacturer	Gigabyte Technology Co., Ltd.
ProductName	G41M-ES2L
SerialNumber	

Additional Hardware Status reports

Scroll to the bottom of the Hardware Status section to access these links to additional reports:

- Test Network - Pings all nodes in the cluster to ensure that all nodes can communicate with each other using TCP/IP and UDP (see details below).
- Test Volumes - Pings the volumes on your local hard drives and provides a response time (in milliseconds).
- Dmesg Dump - Displays the last 1000 messages logged by the Linux kernel reading buffer. These messages can help you troubleshoot and diagnose a Swarm issue when a system panic or error occurs.
- Hwinfo dump (the Linux hardware detection tool output)
- Send Health Report (script that sends the hardware health report to the configured destination)

Test Network

Test Network performs two sets of tests:

- First, it sends 100 UDP multicasts to the cluster and computes the results:
- Which nodes responded
- How many responses returned
- How long the responses took, on average
- Next, it fetches the status page (port 80) via TCP for all of those responding nodes (only once for each node). It tracks the total time for each of those round trips.

The data in the Network Test Results window lets you compare the responding nodes with the list of nodes that you expected to see in the cluster. You can also evaluate UDP packet loss and TCP connectivity within the cluster.

> Important
> If one or more nodes do not appear in the display, you may have a network issue in the cluster.

Viewing and Editing Feeds - Legacy Admin Console

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Viewing Cluster Feeds
- Adding a feed
- Feed Actions
- Troubleshooting Blocked Feeds

A Replication Feed performs replication continuously and adaptively:

- Replicates all of the objects in your source cluster or domain.
- Performs continuous replication to keep up with source cluster intake, within available connectivity and bandwidth to the target cluster.
- Verifies object replication to the targeted cluster as soon as possible.
- Uses an intermittent connection (such as HTTP) to move content from the source to the target cluster.

A Search Feed is an object-routing mechanism in the storage cluster that uses intermittent channel connections to distribute data to Elasticsearch for object metadata searching. The source cluster processes all UUIDs and names stored in the source cluster based on your feed configurations in the Swarm Admin Console Settings page. As objects are added to the cluster, Swarm adds the UUIDs and names to the assigned feed queue and notifies the target Elasticsearch server that feed data is available.

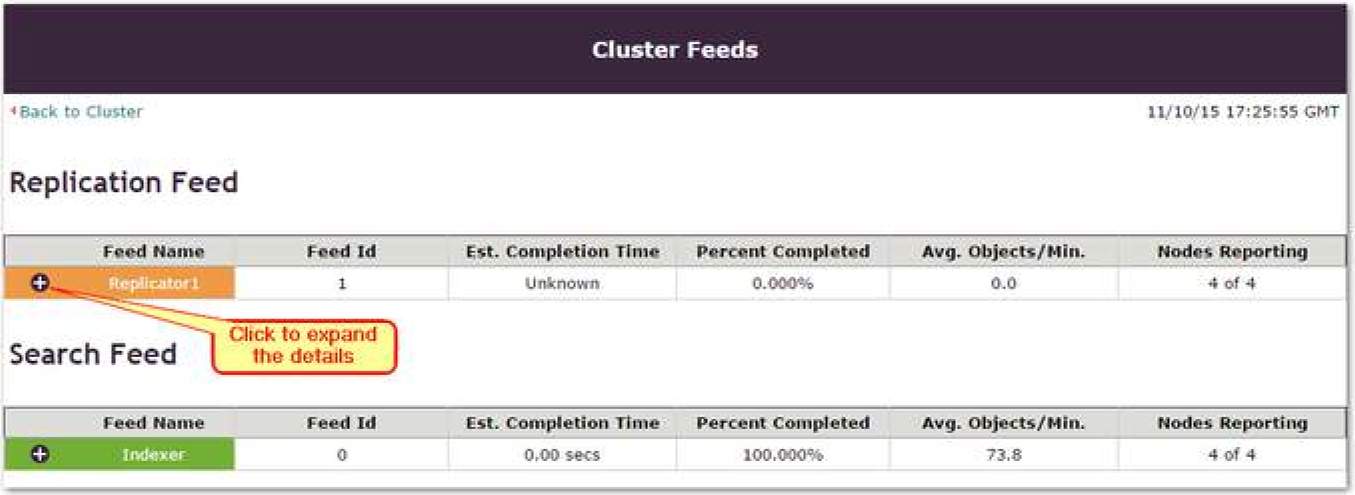
> Note
> You can have multiple search feeds to populate different ES clusters, but only one can be designated as the primary, for searching from Swarm.

Viewing Cluster Feeds

When you click Feeds in the Swarm Admin Console, the Cluster Feeds page appears. Feeds are an object routing mechanism in the storage cluster that uses intermittent channel connections to distribute data to one or more targeted storage clusters.

Using Feeds, you can send:

- Metadata content from the source cluster to the search servers (Search Feed)
- Objects from the source cluster or a particular domain to a targeted cluster (Replication Feed)



Cluster Feeds

‹Back to Cluster

11/10/15 17:25:55 GMT

Replication Feed

Feed Name	Feed Id	Est. Completion Time	Percent Completed	Avg. Objects/Min.	Nodes Reporting
⊕ Replicator1	1	Unknown	0.000%	0.0	4 of 4

Click to expand the details

Search Feed

Feed Name	Feed Id	Est. Completion Time	Percent Completed	Avg. Objects/Min.	Nodes Reporting
⊕ Indexer	0	0.00 secs	100.000%	73.8	4 of 4

Column	Description
Feed Name / Feed ID	The name of the feed and its corresponding number that uniquely identifies the feed.
Estimated Completion Time	The estimated time required to process and clear the current backlog of replication feeds based on the last 60 minutes. This estimate assumes that conditions do not change.
Percent Completed	The dynamic report of progress to completion, as a percentage.
Avg Objects / Min	The average number of objects processed per minute based on the processing rate during the last 60 minutes.
Nodes Reporting	The number of cluster nodes that sent feed reports to be aggregated.

Each feed provides color status indicators that shows its current state. The following table describes the status colors and corresponding states for search and replication feeds.

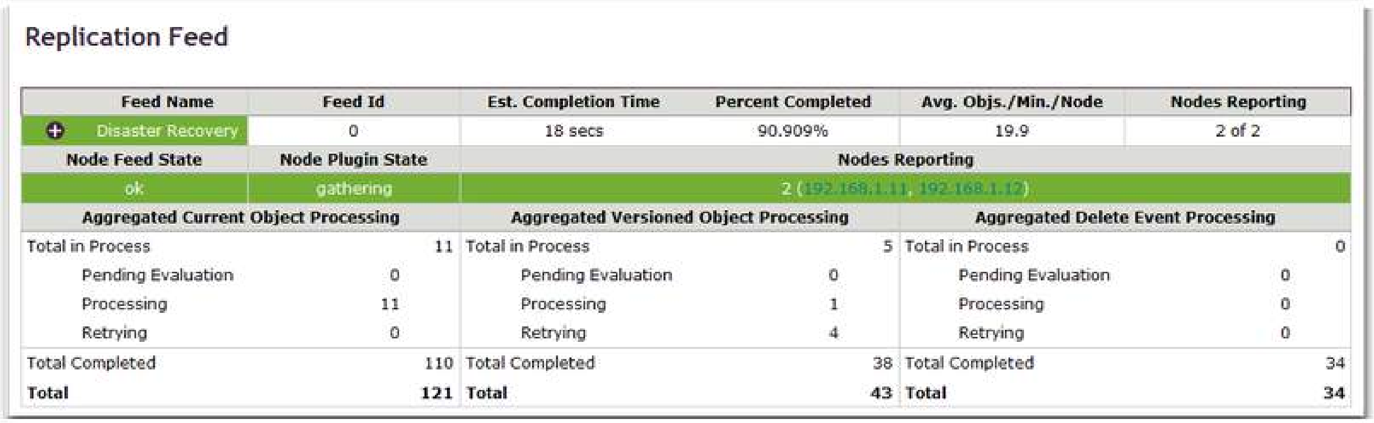
Color	State	Description
Green	OK	The feed replicated successfully to the targeted search server or cluster node.
Gray	Paused	The feed is suspended because of an ongoing fast volume recovery (FVR) issue on the node.
Orange	Blocked	The destination node or cluster is offline or not accepting feeds. • For Search feeds, the search service may be offline. • For Replication feeds, the remote cluster may be offline.
Red	Config Error	The feed is misconfigured.

Viewing a replication feed

The Replication Feed window provides the statistics for a replication feed sent to a targeted Replication cluster. Depending on your configuration, a typical configuration can have more than one replication feed.

When an issue occurs for a particular Replication feed, Swarm logs a critical message and updates the status color and state in the window so you can quickly identify a problem.

When you expand the Replication Feed, the following data appears:



Replication Feed

Feed Name	Feed Id	Est. Completion Time	Percent Completed	Avg. Objs./Min./Node	Nodes Reporting
⊕ Disaster Recovery	0	18 secs	90.909%	19.9	2 of 2

Node Feed State	Node Plugin State	Nodes Reporting
ok	gathering	2 (192.168.1.11, 192.168.1.12)

Aggregated Current Object Processing		Aggregated Versioned Object Processing		Aggregated Delete Event Processing	
Total in Process	11	Total in Process	5	Total in Process	0
Pending Evaluation	0	Pending Evaluation	0	Pending Evaluation	0
Processing	11	Processing	1	Processing	0
Retrying	0	Retrying	4	Retrying	0
Total Completed	110	Total Completed	38	Total Completed	34
Total	**121**	**Total**	**43**	**Total**	**34**

Column	Description and States
Node Feed State	The current feed processing state to the Replication cluster. The state can be: • OK. Operating normally. • Paused. Temporarily paused due to volume recovery. • Blocked. Processing blocked due to a transient condition. The Node Plugin State will indicate Blocked as well. (See below.)

Node Plugin State	The HTTP transaction state to the Replication cluster. The state can be: • Idle. No pending work or replication at this time. • Gathering. Gathering replication entries for a future request. • Replicating. Processing ongoing requests to the destination cluster. • Blocked. Waiting for a transient error to clear. (See below.) • Permanent Error. Permanently stuck due to configuration errors.|
Nodes Reporting	The IP addresses of each reporting node that contributes to the aggregate report. If all target cluster nodes reported successfully, they appear together as a group. If one or more nodes did not report successfully, they appear in separate rows with their corresponding feed state. Each IP address hyperlinks to the SNMP Repository Dump page for the reporting node, which provides node-specific detail for each feed.
Aggregated Current Object Processing	Provides the aggregated statistics for new or updated objects. These statistics include: • Pending Evaluation. Objects that must be evaluated before they are assigned to the replication feed. • Unprocessed. Objects queued for processing by the target cluster. • Successes. Objects replicated by the target cluster. • Retrying. Objects rejected by the target cluster due to a server or network problem. These objects will continue to be transmitted until they are accepted by the target cluster.
Aggregated Versioned Object Processing	Provides the aggregated statistics for versioned objects: • Pending Evaluation. Versions that must be evaluated before they are assigned to the replication feed. • Unprocessed. Versions queued for processing by the target cluster. • Successes. Versions processed by the target cluster. • Retrying. Versions that were rejected by the target cluster, due to a server or network problem. These objects will continue to be transmitted until they are accepted.
Aggregated Delete Event Processing	Provides the aggregated statistics for deleted objects. These statistics include: • Pending Evaluation. Object deletes received but not evaluated by the target cluster. • Unprocessed. Object deletes queued for processing by the target cluster. • Successes. Object deletes replicated by the target cluster. • Retrying. Object deletes that were not processed by the target cluster due to a server or network problem. The object deletes will continue to be transmitted until they are processed by the target cluster.

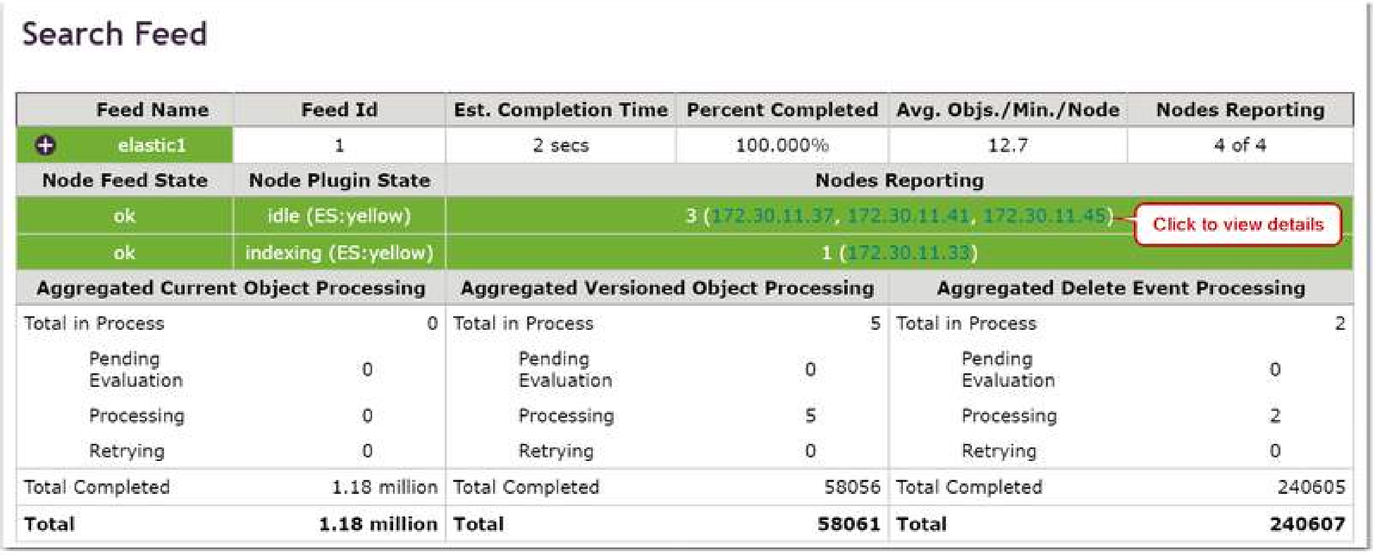
Viewing a search feed

The Search Feed window provides statistics for a search feed sent to a targeted Elasticsearch cluster. A typical configuration will only have one search feed.

When an issue occurs for a particular search feed, Swarm logs a critical message and updates the status color and state in the window so you can quickly identify a problem.

> Note
> Elasticsearch proactively monitors the disk space on its nodes. Elasticsearch suspends itself on nodes as soon as space falls below 5% and automatically unblocks itself when space becomes available, without requiring you to restart Swarm.

When you expand the Search Feed, the following status summaries appear:



Search Feed

Feed Name	Feed Id	Est. Completion Time	Percent Completed	Avg. Objs./Min./Node	Nodes Reporting
⊕ elastic1	1	2 secs	100.000%	12.7	4 of 4

Node Feed State	Node Plugin State	Nodes Reporting
ok	idle (ES:yellow)	3 (172.30.11.37, 172.30.11.41, 172.30.11.45)
ok	indexing (ES:yellow)	1 (172.30.11.33)

Click to view details

Aggregated Current Object Processing		Aggregated Versioned Object Processing		Aggregated Delete Event Processing	
Total in Process	0	Total in Process	5	Total in Process	2
Pending Evaluation	0	Pending Evaluation	0	Pending Evaluation	0
Processing	0	Processing	5	Processing	2
Retrying	0	Retrying	0	Retrying	0
Total Completed	1.18 million	Total Completed	58056	Total Completed	240605
Total	**1.18 million**	**Total**	**58061**	**Total**	**240607**

Column	Description and States
Node Feed State	The current feed processing state to the search cluster nodes. The state can be: • OK. Operating normally. • Paused. Temporarily paused due to volume recovery. • Blocked. Blocked due to a transient condition. The Node Plugin State will indicate Blocked as well. (See below.)
Node Plugin State	The HTTP transaction state to the search server. The state can be: • Idle. No pending work or replication at this time. • Indexing. Processing ongoing requests to the search server. • Blocked. Waiting for a transient error to clear. (See below.) • Permanent Error. Permanently stuck due to configuration errors.

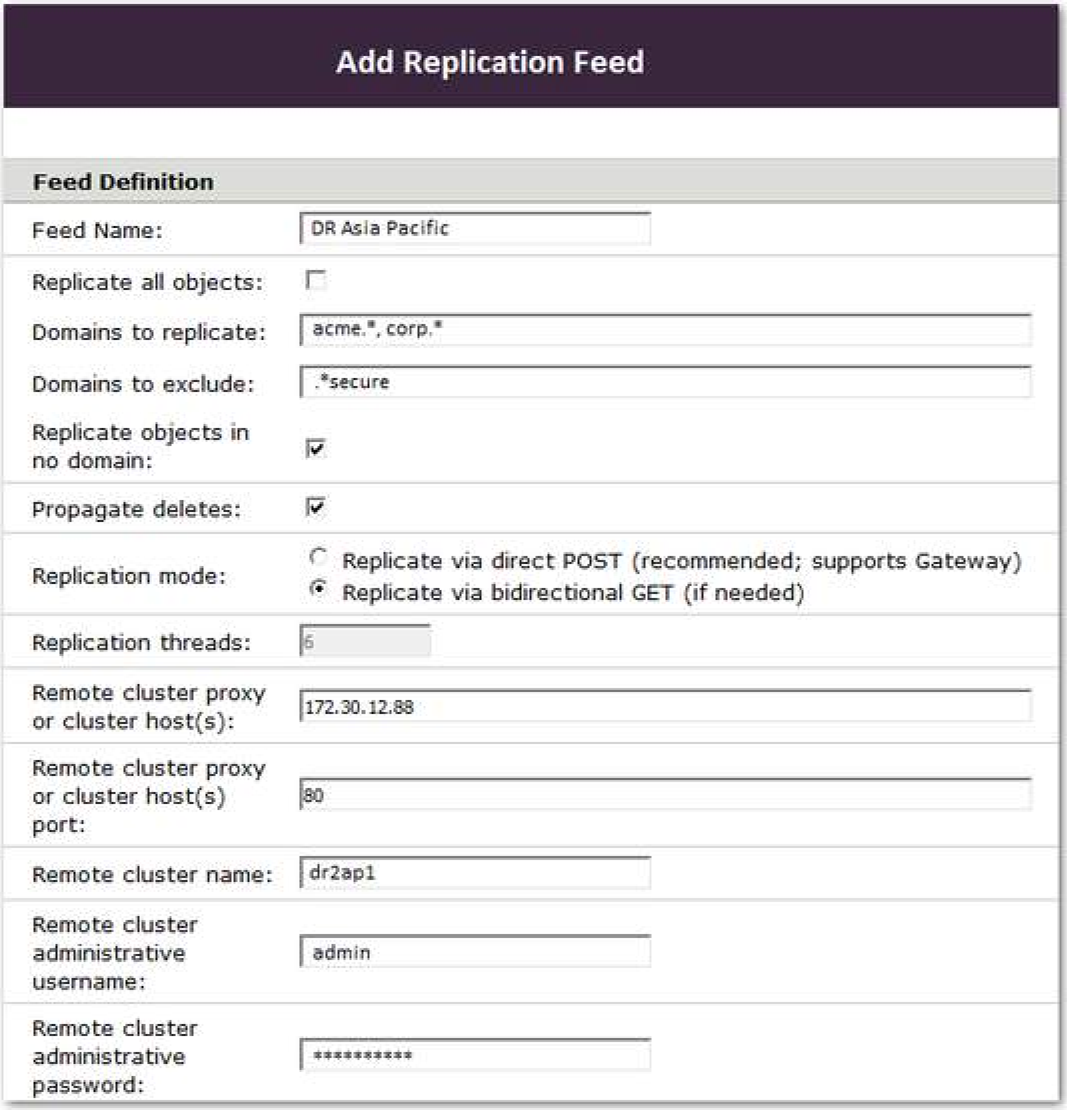
Nodes Reporting	The number of nodes and their corresponding IP address that contributes to the aggregate report. If the search nodes reported successfully, they appear together as a group. If one or more nodes did not report successfully, they appear in separate rows with their corresponding feed state. Each IP address hyperlinks to the SNMP Repository Dump page for the reporting node, which provides node-specific detail for each feed.
Aggregated Current Object Processing	Provides the aggregated statistics for new or updated objects: • Pending Evaluation. Objects that must be evaluated before they are assigned to the search feed. • Unprocessed. Objects queued for processing by the search service. • Successes. Objects processed by the search service. • Retrying. Objects that were rejected by the search service, due to a server or network problem. These objects will continue to be transmitted until they are accepted.
Aggregated Versioned Object Processing	Provides the aggregated statistics for versioned objects: • Pending Evaluation. Versions that must be evaluated before they are assigned to the search feed. • Unprocessed. Versions queued for processing by the search service. • Successes. Versions processed by the search service. • Retrying. Versions that were rejected by the search service, due to a server or network problem. These objects will continue to be transmitted until they are accepted.
Aggregated Delete Event Processing	Provides the aggregated statistics for deleted objects: • Pending Evaluation. Object deletes that do not belong to the search feed. • Unprocessed. Object deletes queued for processing by the search service. • Successes. Object deletes processed by the search service. • Retrying. Object deletes that were not processed by the search service, due to a server or network problem. The object deletes will continue to be transmitted until they are processed.

Adding a feed

You implement Feeds through the Cluster Settings page.

Adding a replication feed

The Add Replication Feed dialog box allows you to enter the configuration settings for a replication feed to a target cluster.



Add Replication Feed

Feed Definition

Feed Name:	DR Asia Pacific
Replicate all objects:	□
Domains to replicate:	acme.*, corp.*
Domains to exclude:	.*secure
Replicate objects in no domain:	☑
Propagate deletes:	☑
Replication mode:	○ Replicate via direct POST (recommended; supports Gateway) ● Replicate via bidirectional GET (if needed)
Replication threads:	6
Remote cluster proxy or cluster host(s):	172.30.12.88
Remote cluster proxy or cluster host(s) port:	80
Remote cluster name:	dr2ap1
Remote cluster administrative username:	admin
Remote cluster administrative password:	**********

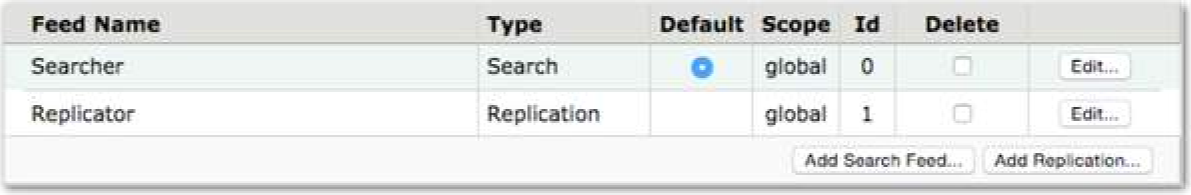
Feed Name	The friendly name that you attach to this feed. This name will appear in the Feeds row in the Settin gs page.
Replicate all objects	Enable to replicate all objects in the source cluster to the target cluster, regardless of domain. If you disable this option, the filtering options become available and must be populated with valid values.

Domains to replicate	Required. Specify one or more domains to include, by name (`hrfoo.example.com, itfoo.example.com`) or by wildcard (`.*foo`). Important: To filter by exclusion only, specify all (`.*`) or else no domains will be allowed. Wildcards Pattern matching follows the Python regular expression (RE) syntax with the exception that the "{m,n}" repetitions qualifier may not be used.
Domains to exclude	Optional. Specify one or more domains to exclude from the set of domains to replicate, by name (`abc123.example.com, abc456.example.com`) or by wildcard (`abc.*`).
Replicate objects in no domain	Optional. Includes unnamed objects that are not tenanted in any domain. If you leave the domain lists above unspecified, enabling this option creates a filtered feed that only replicates these unnamed objects.
Propagate deletes	Enable to keep object deletes synchronized with the source cluster. Disable to prevent objects from being deleted in the target cluster.
Replication mode	Specifies replication via direct POST (recommended) or bidirectional GET. For best performance, choose direct POST replication, which can go through Gateway. GET replication is the legacy method, which may be needed for application compatibility or networking requirements. Switching modes does not require a feed restart. (v9.6)
Replication threads	Replication via direct POST only. The default replication speed (6 simultaneous threads) is best for same-sized clusters with minimal replication backlog. (v9.6) To avoid overwhelming a smaller target cluster, reduce the threads. For faster replication against a backlog, increase the threads temporarily, but be sure to monitor bandwidth and cluster performance, as boosting the speed will stress both clusters.
Remote cluster proxy or cluster host(s)	The IP address of either: • One or more nodes in the target cluster. • A reverse proxy host that routes to the target cluster. To enter two or more node IP addresses, enter each address separated by a comma or spaces.
Remote cluster proxy or cluster host(s) port	Defaults to 80. Lets you specify a custom port for the remote cluster. (v9.6)
Remote cluster name	The configuration setting for your target cluster (for example, the cluster.name value in the .cfg file of the target cluster).

Remote cluster administrative user name	The administrative user name of the target cluster. Enter a value in this field only if the remote cluster user name is different from the source cluster name in the same realm.
Remote cluster Administrative password	The administrative password of the target cluster. Enter a value in these fields only if the remote cluster password is different from the password on the source cluster in the same realm.

To add a replication feed:

1. Open the Swarm Admin Console and click Settings.
2. In the Feed Name row, click Add Replication.
3. When prompted for authentication, enter your administrator name and password.
4. In the Add Replication Feed window, complete the fields as described above.
5. Click Add. Your new feed appears in the Feed Name row in the Cluster Settings page and propagates to the targeted nodes in your cluster within 60 seconds.

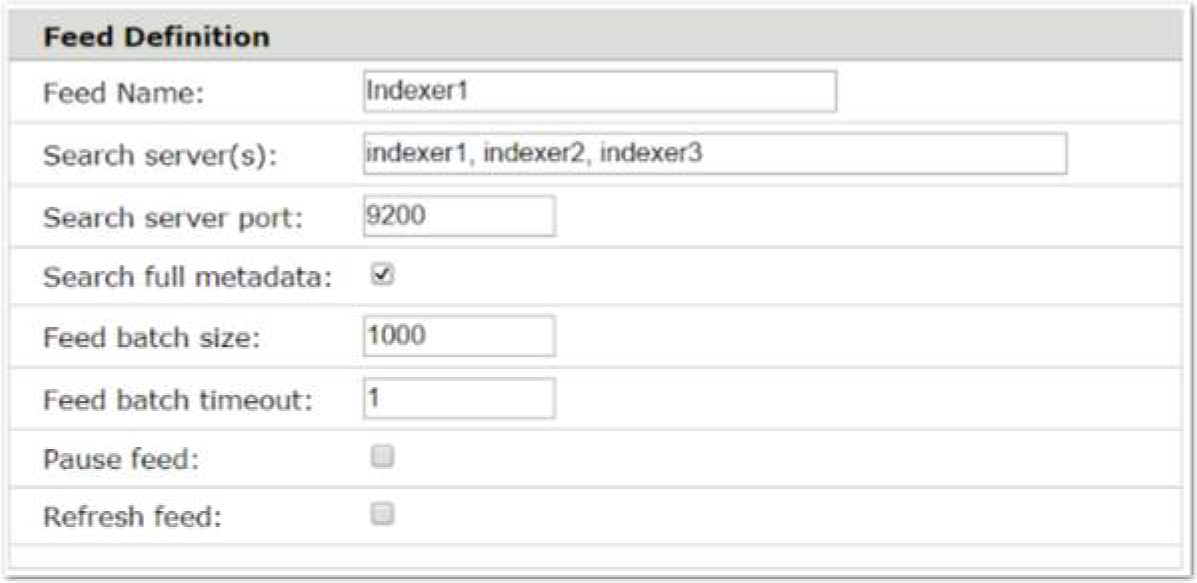


Feed Name	Type	Default	Scope	Id	Delete	
Searcher	Search	●	global	0	□	Edit...
Replicator	Replication		global	1	□	Edit...

6. Click Update. A Success dialog box appears.
7. Click Close.

Adding a search feed

The Add Search Feed dialog box allows you to enter the configuration settings for a search feed to the Elasticsearch server.



Feed Definition

Feed Name:	Indexer1
Search server(s):	indexer1, indexer2, indexer3
Search server port:	9200
Search full metadata:	☑
Feed batch size:	1000
Feed batch timeout:	1
Pause feed:	□
Refresh feed:	□

The following table describes the data entry fields in the dialog box.

Field	Description
Feed name	The name you attach to the feed.
Search server(s)	The IP addresses or server names that are resolvable by DNS. If you enter more than one server, separate each server name with a space. DNS must be configured on both the source and target clusters.
Search server port	The default port for a host.
Search full metadata	Enabled - Swarm indexes all object metadata, including baseline and customer metadata fields. Disabled - Swarm indexes only the baseline metadata fields. See Metadata Field Matching for a list of baseline and custom fields.
Feed batch size	The maximum number of objects that would be sent concurrently to be processed. The default is 100.
Feed batch timeout	The maximum amount of time (in seconds) before a batch is resent to be processed after a timeout. The default is 1.

To add a search feed

1. Open the Swarm Admin Console and click Settings.
2. In the Feed Name row, click Add Search.
3. When prompted for authentication, enter your administrator name and password.
4. In the Add search feed window, complete the fields as described above.

5. Click Add.
   Your new feed appears in the Feed Name row in the Cluster Settings page and propagates to the targeted nodes in your cluster within 60 seconds.
6. Click Update.
   A Success dialog box appears.
7. Click Close.

Feed Actions

Deleting a replication feed

When you delete a feed, it frees source cluster resources. This process does not affect the objects previously pushed to the target cluster.

To delete a feed:

1. In the Swarm Admin Console click Settings.
2. In the Cluster Settings window, locate the Feed Name section.
3. Identify the feed you want to delete and select the corresponding Delete checkbox.
4. Click Update.
   A Success dialog box appears.
5. Click Close.
   The deleted feed is removed from the remaining cluster nodes within 60 seconds.

Editing a search feed

To edit a feed

1. Open the Swarm Admin Console, and click Settings.
2. In the Cluster Settings window, locate the Feed Name section.
3. Select the feed you want to edit and click Edit.
4. In the Edit search feed window, make any changes as needed to the appropriate fields.
5. Set the Refresh feed option based on your Elasticsearch server configuration:
   - Select Refresh feed to hydrate a new search server.
   - Deselect Refresh feed to update a running feed on a current search server.
6. Click Update.
   Your updates appear in the Cluster Settings window and propagate to the remaining cluster nodes within 60 seconds.

Pause feed

When you edit a feed, you can pause the feed or unpause it by toggling the checkbox option.

For example, when you are upgrading your Elasticsearch cluster, you would pause the search feed before stopping the Elasticsearch service in your search cluster.

> See Installing Elasticsearch.

Refresh feed

As objects are written or updated, their metadata is sent to the search servers in near real-time (NRT). Any objects that cannot be processed immediately are retried each HP cycle until they succeed, at which point they are marked as complete and are never resent. If a data loss failure occurs on your Elasticsearch servers and you cannot restore from backup, you can refresh the feed, which will verify and rehydrate all of the metadata content.

When you edit a feed, you choose either to refresh all of the search data for the feed or to add your changes to the running feed, without data verification.

- Enabled - Resends all object metadata to the Elasticsearch server. If an Elasticsearch index for the cluster does not exist, it will be created. To recreate an existing index (such as for case-insensitive searching where case-sensitive was previously used), drop the existing index before refreshing the feed. For example, if you installed a new Elasticsearch server in your domain or are moving it to another domain, select Refresh feed to reverify the feed content on the new or moved server.

  > Note
  > Expect lower performance while the feed rebuilds the metadata index.

- Disabled - Adds your changes to the running feed without resending previously processed objects. For example, if you are updating your current search service, you can deselect Refresh feed to add your changes to the running feed.

> Best practice
> Swarm never resends prior deletions of unnamed objects to the search service, and it resends prior deletions of named objects only within 14 days of the deletion. If you need to purge a large number of deleted objects, drop the search index before refreshing the feed.

Deleting a search feed

When you delete a feed, it frees source cluster resources. This process does not affect the objects previously pushed to the search server.

> Important
> If you delete the default search feed, be sure to define or select another search feed and mark it as default, to support Elasticsearch queries.

To delete a feed

1. Open the Swarm Admin Console, and click Settings.
2. In the Cluster Settings window, locate the Feed Name section.
3. Identify the feed you want to delete and select the corresponding Delete checkbox.
4. In the Security dialog box, enter your administrator name and password and click OK.
5. Click Update. A Success dialog box appears.
6. Click Close. Your changes propagate to the targeted nodes in your cluster within 60 seconds.

> You may also wish to delete the search data previously sent by the feed.

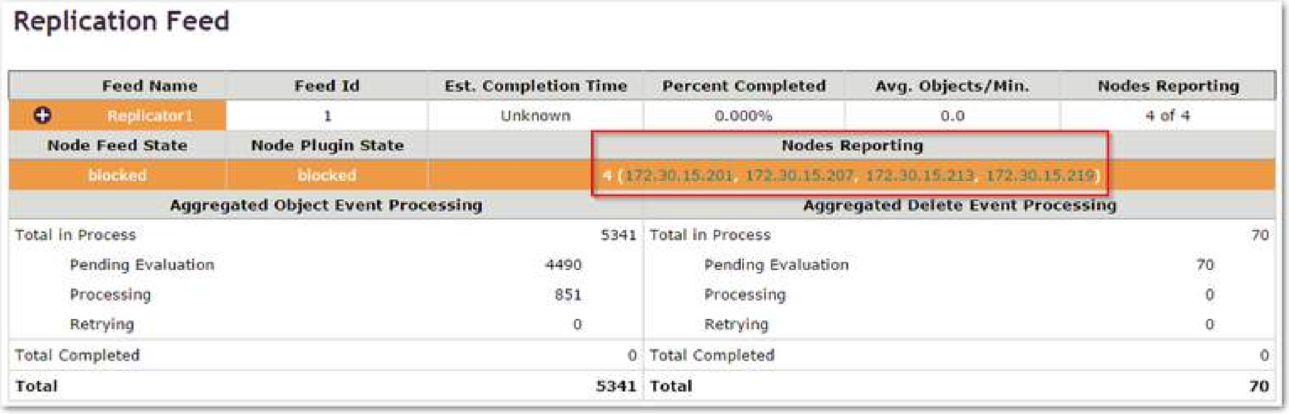
Troubleshooting Blocked Feeds

If the Node Feed State indicates blocked , you can troubleshoot the feed state to the Replication cluster by reviewing the Feeds Table data in the SNMP Repository Dump page for each reporting cluster node.

> For more information about the SNMP Repository Dump tables, see the SNMP MIB Reference file included in the top level of the Swarm product distribution ZIP file.

To troubleshoot feeds:

1. Log in to the Swarm Admin Console as an Administrator.
2. In the console, click Feeds .

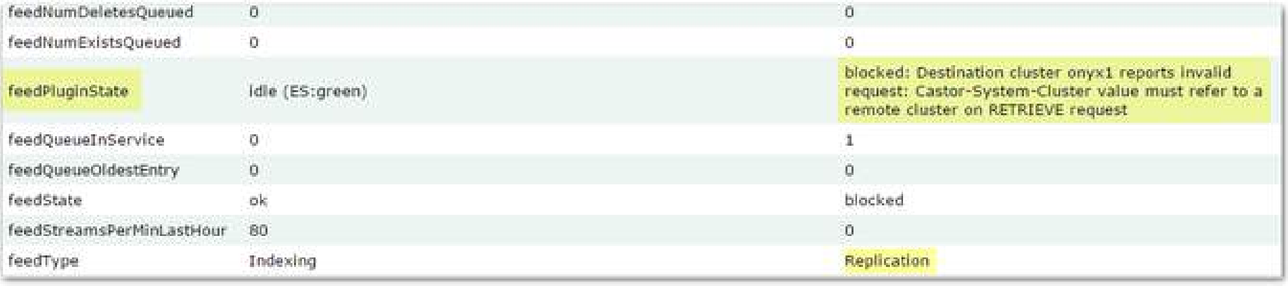


Replication Feed

Feed Name	Feed Id	Est. Completion Time	Percent Completed	Avg. Objects/Min.	Nodes Reporting
⊕ Replicator1	1	Unknown	0.000%	0.0	4 of 4
Node Feed State	Node Plugin State		Nodes Reporting		
blocked	blocked		4 (172.30.15.201, 172.30.15.207, 172.30.15.213, 172.30.15.219)		
Aggregated Object Event Processing			Aggregated Delete Event Processing		
Total in Process		5341	Total in Process		70
Pending Evaluation	4490		Pending Evaluation	70	
Processing	851		Processing	0	
Retrying	0		Retrying	0	
Total Completed		0	Total Completed		0
Total		**5341**	**Total**		**70**

- In the Cluster Feeds window, the orange status highlights the blocked node feed and plugin states for the Feed.
- The Nodes Reporting box lists the IP addresses for any replication cluster nodes that are offline or not accepting feeds.

3. In the Nodes Reporting box, click a reporting node IP address.
   The Feed Table in the SNMP Repository Dump page appears for the selected node.
4. Review the feedPluginState status to identify the blockage.
   Example:



feedNumDeletesQueued	0	0
feedNumExistsQueued	0	0
feedPluginState	idle (ES:green)	blocked: Destination cluster onyx1 reports invalid request: Castor-System-Cluster value must refer to a remote cluster on RETRIEVE request
feedQueueInService	0	1
feedQueueOldestEntry	0	0
feedState	ok	blocked
feedStreamsPerMinLastHour	80	0
feedType	Indexing	Replication

```
feedPluginState          blocked: Destination cluster onyx1 reports
invalid request: Castor-System-Cluster value must refer to a remote
cluster on RETRIEVE request
```

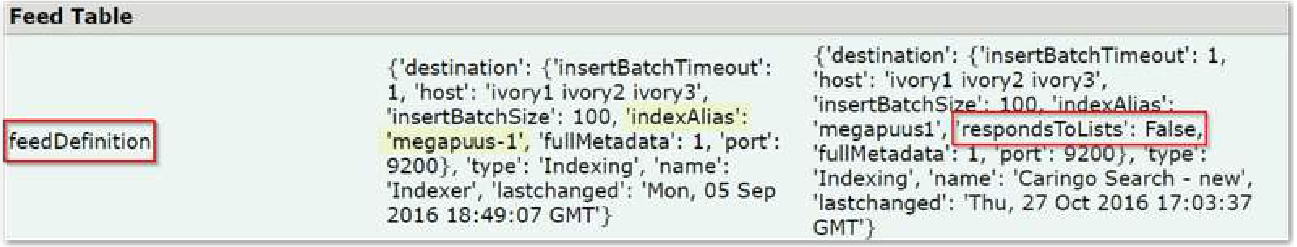
5. Repeat to troubleshoot the remaining blocked IP addresses.

Identifying the Primary Search Feed

If you do not have access to the Swarm Storage UI, you can find Swarm's alias name for your primary search index using one of these methods:

Legacy Admin Console - View the SNMP Repository Dump to find out what search feeds are defined:

1. Open the dump via the legacy Admin Console: `http://{host_or_ip}:90/snmp_dump`
2. Click Feed_Table and look at the row for feedDefinition.
3. If you have more than one feed (column), the primary (default) feed is the one that does not have 'respondsToLists': False:



Feed Table		
feedDefinition	{'destination': {'insertBatchTimeout': 1, 'host': 'ivory1 ivory2 ivory3', 'insertBatchSize': 100, 'indexAlias': 'megapuus-1', 'fullMetadata': 1, 'port': 9200}, 'type': 'Indexing', 'name': 'Indexer', 'lastchanged': 'Mon, 05 Sep 2016 18:49:07 GMT'}	{'destination': {'insertBatchTimeout': 1, 'host': 'ivory1 ivory2 ivory3', 'insertBatchSize': 100, 'indexAlias': 'megapuus1', 'respondsToLists': False, 'fullMetadata': 1, 'port': 9200}, 'type': 'Indexing', 'name': 'Caringo Search - new', 'lastchanged': 'Thu, 27 Oct 2016 17:03:37 GMT'}

4. For the primary feed, note the value for 'IndexAlias' and enter this in the NFS export definition.

curl - Query to find out what search feeds are defined:

```
curl http://{host_or_ip}:91/api/storage/feeds?pretty=true
```

If more than one is listed, query to find out which one is the primary (default) feed:

```
curl http://{host_or_ip}:91/api/storage/feeds/0?pretty=true | grep
"isDefault|indexAlias"
curl http://{host_or_ip}:91/api/storage/feeds/1?pretty=true | grep
"isDefault|indexAlias"
```

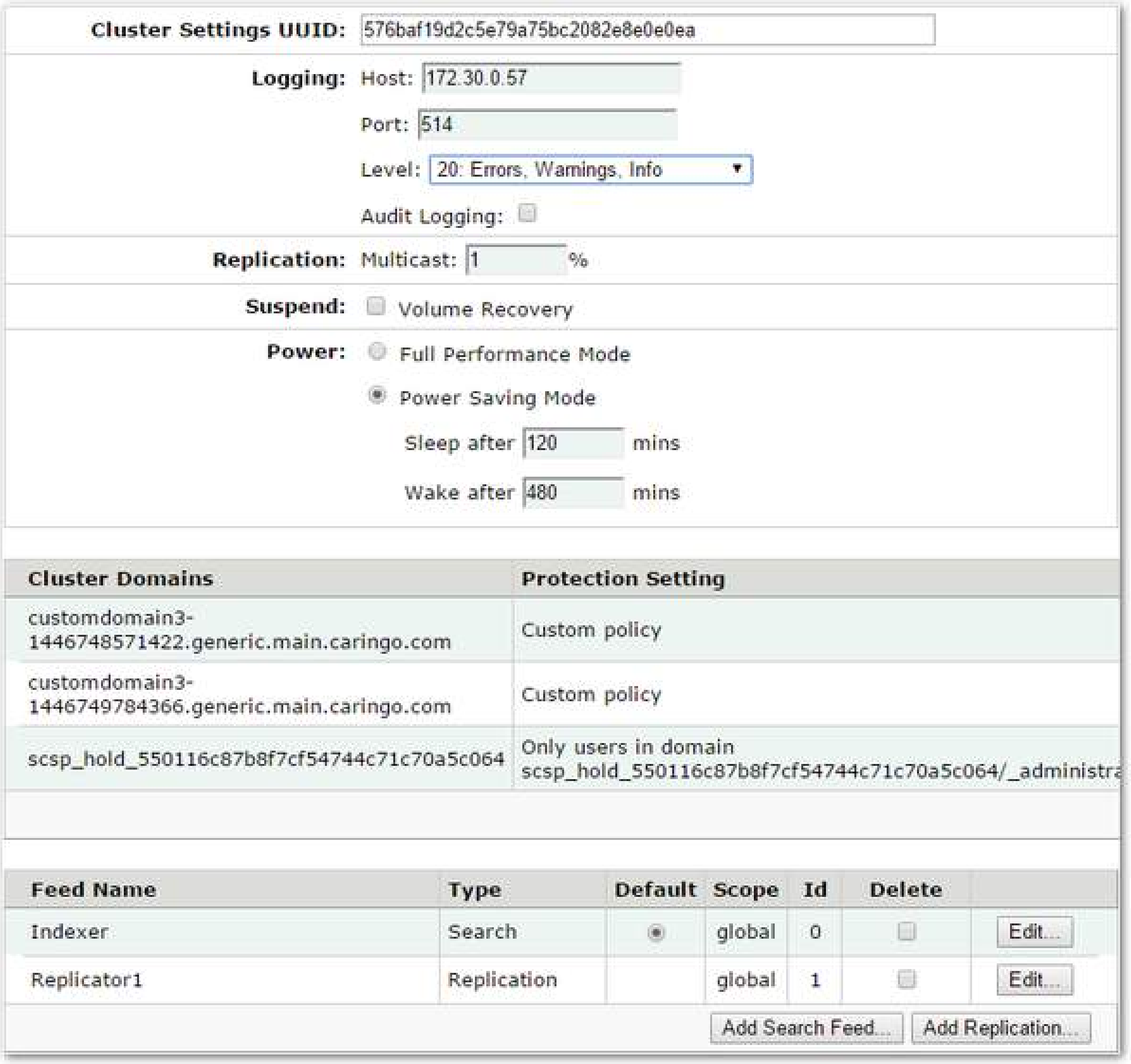
Using Cluster Settings - Legacy Admin Console

> Deprecated
> The Legacy Admin Console (port 90) is still available but has been replaced by the Swarm Storage UI. (v10.0)

- Changing the cluster name
- Changing the multicast address
- Logging setting
- Replication setting
- Suspend setting
- Power setting
- Cluster Domains setting

- Feeds setting

When you click Settings in the Swarm Admin Console, the Cluster Settings dialog box appears, enabling you to make runtime changes to several configuration settings in your cluster. When you save your settings for the first time, the Cluster Settings UUID field displays the universal unique identifier (UUID) of the aliased object that contains the cluster settings. All changes will persist in the cluster across reboots.



Cluster Settings UUID: 576baf19d2c5e79a75bc2082e8e0e0ea

Logging: Host: 172.30.0.57
Port: 514
Level: 20: Errors, Warnings, Info
Audit Logging: ☐

Replication: Multicast: 1 %

Suspend: ☐ Volume Recovery

Power: ○ Full Performance Mode
● Power Saving Mode
Sleep after 120 mins
Wake after 480 mins

Cluster Domains	Protection Setting
customdomain3-1446748571422.generic.main.caringo.com	Custom policy
customdomain3-1446749784366.generic.main.caringo.com	Custom policy
scsp_hold_550116c87b8f7cf54744c71c70a5c064	Only users in domain scsp_hold_550116c87b8f7cf54744c71c70a5c064/_administra

Feed Name	Type	Default	Scope	Id	Delete	
Indexer	Search	●	global	0	☐	Edit...
Replicator1	Replication		global	1	☐	Edit...

Caution

Do not set cluster-wide persisted settings in the individual node configuration files because these values must be the identical for the entire cluster. Any values specified in a node configuration file are overwritten by the cluster settings in the Swarm Admin Console.

Prior versions

As of Swarm version 6.0, the cluster settings are automatically propagated to all nodes in the cluster. Adding the cluster settings UUID to your node or cluster configuration file is not required.

The first time you change a cluster-wide setting, the cluster settings UUID is automatically generated and propagated to all nodes in the cluster. You can change cluster-wide settings in the Swarm Admin Console by manually or programmatically adding a domain (see Managing Domains) or using snmpset (see Using SNMP with Swarm).

If you upgraded from an earlier Swarm version, the clusterSettingsUUID setting value in the node or cluster configuration file is used when the node initially boots in version 6.0. After at least one node has booted to version 6.0, you can remove the clusterSettingsUUID setting from your node or cluster configuration file.

Changing the cluster name

You can change the cluster name in the cluster.name configuration setting located in the node or cluster configuration file. Follow the guidelines below.

> Important
>
> Neither the cluster name nor the cluster multicast address can be changed at runtime.

- Set the cluster.name configuration setting value before you change any cluster settings. After the cluster settings UUID is set, do not change the value.
- If you set the cluster.name and change it or have no cluster.name set and add one after you have a cluster settings UUID, you may encounter conflicts in your settings UUID. Contact Support for instructions on resetting your cluster settings UUID.
- If you move a volume between clusters, the volume will keep its original cluster settings and cluster.name setting value. To overwrite the original settings with the new cluster settings, you must insert the volume into its new location while the remaining nodes are running. If all volumes are mounted at the same time, the settings from the originating cluster will be detected first and become "master" in the new cluster.

Changing the multicast address

The cluster nodes use a single multicast IP address to broadcast cluster-wide messages. To change the IP address, modify the cip.group configuration setting in the node or cluster configuration file.

> Important
>
> Neither the cluster name nor the cluster multicast address can be changed at runtime.

Logging setting

You can update the logging Host, Port, and Level options to temporarily or permanently redirect the cluster logs to a different location or log level. This can be useful when troubleshooting an issue in the cluster that requires a more granular report of cluster activity.

In addition, you can check the Audit Logging checkbox to send audit level events to the syslog, independent of the log level.

Replication setting

This option allows you to set the frequency of multicast broadcasts within the cluster.

The higher the percentage you enter in the Multicast % field, the more frequently the cluster will communicate UUIDs to the cluster nodes and other services.

Suspend setting

Swarm will automatically initiate recovery of replicas and erasure-coded segments known to be on a cluster volume that is no longer present in the cluster. Two recovery processes, failed volume recovery (FVR) and erasure coding recovery (ECR), are started for every volume detected as missing. Swarm will announce both when it starts each process as well as when each completes, which may be relatively quick if there are no replicas or segments to recover.

The Suspend option allows you to temporarily suspend both volume recovery processes in the cluster for situations where data is not actually at risk, but a network or power outages have taken one or more nodes (or an entire sub-cluster) offline. Suspending volume recovery by selecting the Volume Recovery check box prevents cluster activity churn and reduces the risk of capacity issues due to over-replication during an outage or upgrade.

> Important
>
> Do not suspend volume recovery indefinitely, as this would hamper one of Swarm's primary data protection layers.

Power setting

This option allows you to select either a full performance mode or a power-saving mode.

- Full Performance Mode disables the power-saving settings. All nodes remain active (never idle) and volumes become idle after at least six minutes of no I/O activity.
- Power Saving Mode enables the Sleep After and Wake after settings.
- Sleep After (the `power.sleepAfter` setting) sets the length of time without SCSP activity before health processing is paused and the node displays as Idle in the Swarm Admin Console.
- Wake After (the `power.wakeAfter` setting) sets the length of time a node remains idle before the health processor is reactivated.

> Best practice
>
> If your cluster is in constant use (24x7) or if uninterrupted feed restarts are critical for your operations, use Full Performance Mode.

Cluster Domains setting

Cluster domains are secure domains that live entirely within a Swarm storage cluster. Similar to a storage facility containing multiple storage units, domains contain multiple buckets that allow you to store unstructured data objects into specific categories, such as documents, photos, and videos.

> See Managing Domains.

Feeds setting

Feeds is an object routing mechanism in the Swarm storage cluster that uses intermittent channel connections to distribute data to one or more targeted Elasticsearch servers or target clusters. The source cluster processes all UUIDs and names stored in the source cluster based on your feed configurations. This section lists the feeds that are configured for the source cluster.

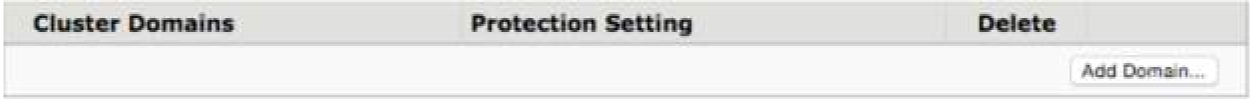
> See Viewing and Editing Feeds for configuring replication and search feeds.

Managing Domains - Legacy Admin Console

> Gateway
> Skip this section if your client applications are accessing your cluster via the Content Gateway.

To add, edit, or delete a domain from the Swarm Admin Console:

1. Open the Swarm Admin Console, and click Settings.
2. In the Cluster Settings page, make sure that the Cluster Domains box is empty if no domains are configured.



3. To add a domain, click Add Domain.
   To edit a domain, click Edit next to its name.
   To delete a domain, select the check box next to its name, click Delete, and click Submit .

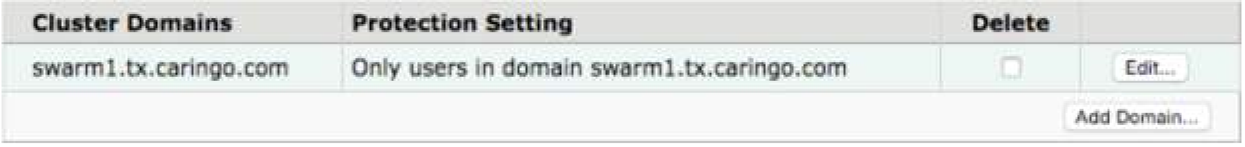
> Note
> If you delete a domain that contains buckets without including the recursive query argument, the buckets and any objects they contain are not deleted, but they are inaccessible. To work around this issue, see Restoring Domains and Buckets.

4. Enter or edit the following information:

Option	Description
Domain Name field	Enter a fully qualified IANA-compliant name to identify this domain (for example, cluster.example.com). The domain name must be unique among all clusters you manage. If you did not configure a domain name that matches your cluster's name, the cluster name displays in this field. Creating a domain with the same name as the cluster sets up a default cluster domain. See the Naming Rules. To rename an existing domain, use the Swarm Admin Console.
Protection Setting	Determines what users are authorized to POST to the domain (such as buckets and unnamed objects). Click one of the following: • All Users. No authentication required. Any user can create buckets or unnamed objects in the domain without authentication. • Only users in this domain. Enables users in the user list associated with this domain to POST to the domain. • Only users in domain. Enables users in the user list associated with the specified domain to POST to this domain.

Domain Managers	Manages user lists in the domain. For help creating user lists, contact your Support representative. To add a new administrator, click Add Domain Manager. To edit an existing manager, click Edit next to the administrator's manager. Important Domain manager names consist of ASCII characters only and cannot include a colon character (:). See the Naming Rules . Tip If Custom Policy displays for Protection Setting, a domain administrator has manually altered the protection settings. If you are trying to troubleshoot an issue with users being able to access objects in a domain, try setting the protection setting back to its default setting.

5. If you are adding or editing a domain manager, enter or edit the following information:
   - User ID. Enter a name to identify the domain manager. Domain manager names consist of ASCII characters only and cannot include a colon character (:).
   - Password. Enter a password for the domain manager.
6. Click Submit.
7. If prompted, enter an administrative user name and password.
   This is how a domain appears in the Cluster Settings page:



Cluster Domains	Protection Setting	Delete	
swarm1.tx.caringo.com	Only users in domain swarm1.tx.caringo.com	□	Edit...
			Add Domain...

Note

The following error indicates that the previous domain name has not expired from the content cache: `ERROR! realm name 'domain-name/_administrators' already exists.`

For example, if a cluster administrator renamed cluster.example.com to domain.example.com and you attempted to create cluster.example.com before the name has expired from the content cache, you would see this error. In this case, wait several minutes and try again.

Swarm Content Gateway

- Content Gateway Concepts
- Content Gateway Authentication
- Gateway Operations
- Replicating Domains to Other Clusters
- Content Metering
- Gateway Troubleshooting

Content Gateway Concepts

Content Gateway is the key to implementing a cloud storage service that is both successful and secure. Combined with Swarm, Content Gateway provides everything needed to deliver private cloud storage services secure within your data center or a public cloud storage service that can compete with Amazon S3. These are essential features of Content Gateway:

- Works with existing authentication systems. Use LDAP, Active Directory, and Linux PAM authentication for integration into existing corporate identity management systems. Additionally, the system supports token-based authentication for pre-validated access with an optional automatic expiration.
- Granular control of content access. A robust access control mechanism allows for coarse to fine-grained control over access to content within the system.
- Robust administration. Administration is managed through the web-based Content Portal or programmatically through an open and documented management API.
- Metering of usage. Provides historical records of storage and bandwidth usage within the system. The metering data is presented to admins and end-users through the graphical charting in the Content Portal and made available through the management API for integration with external systems and applications.
- Audit logging. Administrators can access a data feed that includes details about the all requests to the system that includes the success/rejection response, the user making the request, and the duration of the request. The data feed is designed for machine parsing so that it can be used for access audits, API request analysis, and SLA reporting.
- Flexible naming schemes with no bucket limits. Easy storage management and flexible bucket naming is enabled by lightweight tenant and domain creation and allocation.
- Uses your domain. Access and deliver content using your corporate domain name. Virtual hosting of buckets is also supported through the S3 API.
- Amazon S3 API support. Provides instant compatibility with a broad range of applications and libraries that use Amazon's S3 service.
- Role of the Gateway
- Gateway Architecture
- Content Gateway Components
- Application Concepts
- Service Proxy

Role of the Gateway

The Content Gateway is a lightweight, web-scale software application used by organizations who need to deploy massively scalable, secure, multi-tenant object storage clouds.

The Gateway works as a reverse-proxy in front of the Swarm storage cluster and extends the RESTful object storage API with the following capabilities:

- S3 object storage protocol
- User and group authentication with external identity management systems
- Access control policies
- Automatic metadata transformation
- Usage metering
- Audit logging

Content Gateway also serves enterprises who need to provide a private storage cloud for their business units and

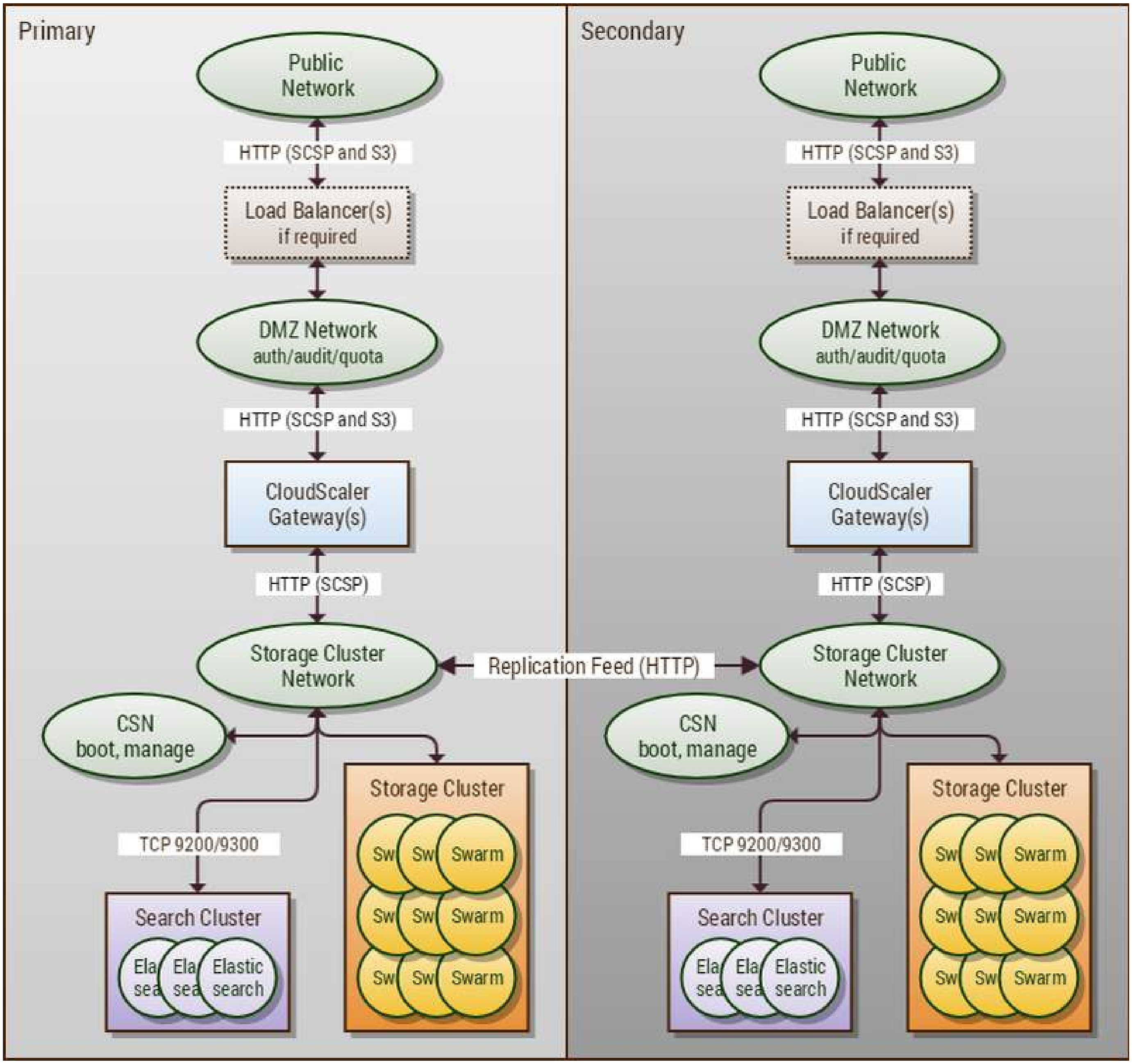
Managed Service Providers and Cloud Service Providers that want to offer public cloud storage as a service.

Gateway Architecture

The Content Gateway involves the following components:

- Client applications that access the object storage cloud via RESTful HTTP calls
- Optional front-end load balancing and/or firewall appliances
- Content Gateway server(s)
- Elasticsearch server(s)
- Swarm Storage cluster

This is an example deployment architecture for the Content Gateway components:



Content Gateway Components

The Content Gateway platform architecture is comprised of hardware and software components.

- Client applications and users
- Load balancer
- Protocol personalities
- Gateway
- Metadata search servers

- Swarm storage cluster
- Remote replication

Client applications and users

Client applications use the Gateway over a network and communicate via the SCSP storage protocol or another protocol such as S3 that is translated by a protocol personality. Client applications include common web browsers such as Firefox, Chrome, or Safari or they can be software from third-party ISVs or custom in-house software developed by customers. Users are people utilizing client application software that communicates with Gateway. When this documentation refers to users accessing Gateway, the implication is that they are making use of one or more client applications to interact with Gateway.

Load balancer

When deploying more than one Gateway, load balancer appliances are a common method for automatically distributing client requests across all Gateways and for excluding Gateways that are offline due to failure or maintenance. Load balancers may optionally implement upstream features such as SSL/TLS end-point termination, protocol firewall rules, quality of service, and geographic traffic management.

The Gateway appears as a normal web proxy to the upstream load balancers and, since the Gateways are stateless, it is not necessary to implement session affinity on the load balancers. Load balancing schemes such as weighted round-robin that prefer to dispatch to the most responsive Gateways are a good choice.

The load-balancing layer is optional for Gateway, and, while hardware appliance load balancers can be well-suited for sites with heavy traffic and sophisticated operational requirements, it is also possible to implement this layer using virtual machines or modest hardware running open-source software. For example, the Pound reverse proxy running on Linux provides transport encryption and load balancing with Layer 7 inspection capabilities.

Protocol personalities

Protocol personalities are optional protocol translators that allow client applications to communicate using a different storage protocol than SCSP. By translating communications in this manner, all client applications, regardless of the protocol they use, share the same content in the back-end cluster and they utilize a common authentication/ACL scheme. An analogy for this universal storage protocol access is ODBC for database communications.

The SCSP and S3 protocol handling is implemented natively within the Gateway. Third-party or user-developed personalities may also be added to, or in front of, the Gateway server.

Gateway

The Gateway is a value-added front-end for the Swarm storage cluster. At its core, it is a stateless, reverse proxy that can be deployed in an n+1 configuration for horizontal scaling and high-availability. The value-added features provided by the Gateway enhance the Swarm SCSP client protocol and provide storage management and protection for the back-end storage cluster.

These value-added features include:

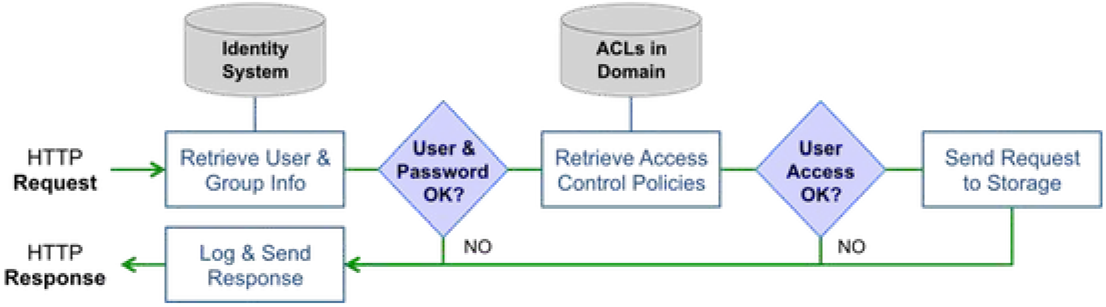
- Authentication for users and access control for content
- Usage metering of storage and bandwidth
- Audit logging for client operations
- Automatic object metadata transformation rules
- Cluster node pool management for load balancing and handling offline nodes
- Reverse proxy to handle SCSP redirects locally to optimize and simplify client communications
- Token-based authentication

- Multi-part MIME uploads

The Gateway is a Java software component that runs within a Jetty servlet container and provides front-end HTTP web services to client applications. The Gateway servers are typically deployed with dual-homed network interfaces to provide proxy services between a front-end client network and a private, back-end storage network.

The Gateway provides protocol isolation and performs SCSP protocol inspection for the incoming client storage requests before passing them along to the storage cluster. This allows for, among other things, the implementation of business rules for content metadata, access control and administrative override for tenant content, and audit/billing event logging.

The Gateway implements the following authentication and request authorization logic:



When using the Gateway front-end for Swarm, the following Swarm features cannot be used when communicating throu gh the Gateway:

- Integrity Seal hash-type upgrades
- Trailing Content-MD5 headers
- [deprecated] Swarm legacy auth/auth mechanism
- Swarm legal hold mechanism

Metadata search servers

Elasticsearch servers provide a NoSQL data query engine that enables metadata searching with the Swarm storage cluster. The query engine software allows for n+1 deployments that provide horizontal scalability for load sharing and high availability.

> See Elasticsearch for Swarm.

Swarm storage cluster

The Swarm storage cluster provides scalable, object storage engine for the Gateway platform. The storage engine consists of standard x86 hardware that manage and protect storage for multiple tenants.

> See Storage Implementation and Swarm Storage Cluster.

Remote replication

The Swarm replication feeds between clusters can be used with Gateway in the following deployment scenarios:

- Clusters communicate directly with each without passing through Gateway
- Gateway acts as a front-end, reverse proxy for its cluster

Direct communication between the clusters happens through internal routing rules between the storage networks or over a VPN connection between the storage networks. The key aspect of this communication is that no inter-cluster traffic passes through the Gateway.

If the Gateway is to act as a front-end, reverse proxy for a storage cluster that is the target of a Swarm replication feed from another cluster, you must configure the `allowSwarmAdminIP` setting in the `[scsp]` section of the Gateway's configuration file. The value will be the IP address list or prefix of every replication source that will contact this Gateway.

Application Concepts

The Gateway offers developers an integration platform that adds many valuable features to the native Swarm storage API. These features include:

- A multi-tenant framework that provides several levels of control and delegation
- Choice of two object storage APIs: SCSP, S3
- A service provisioning and management API

Object storage APIs

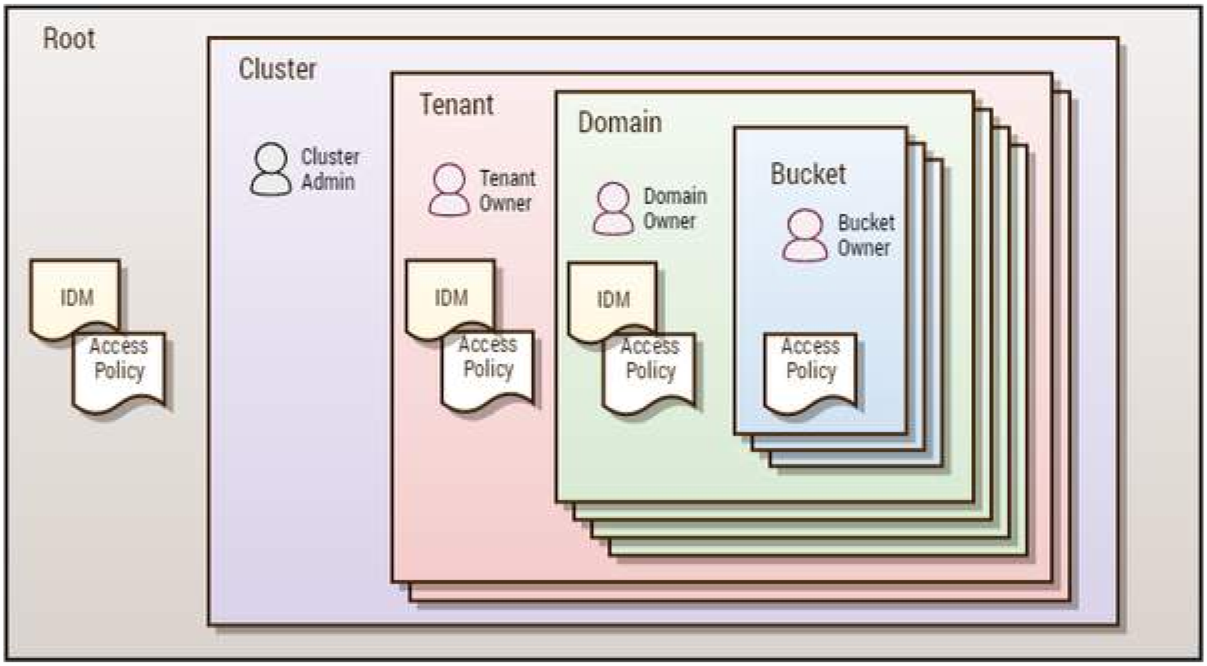
Gateway gives developers the freedom to choose between the Swarm Storage SCSP object storage protocol and the A mazon S3 object storage protocol. Gateway allows both of these protocols to share the back-end Swarm cluster and even the same content. Additionally, Gateway provides enhancements to both object storage protocols that allow for geographically distributed, multi-tenant storage clouds.

S3 and SCSP are the Storage APIs offered through the Gateway. Developers accustomed with Amazon S3 development can continue to use their tools, libraries, and experience and immediately begin using Swarm in their own environment. The Swarm SCSP object storage protocol offers advantages over Amazon S3 in the area of content protection controls, time-based content policies, metadata searching capabilities, and an additional object type: unnamed objects.

- The SCSP object storage protocol is documented in Storage SCSP Development.
- The S3 object storage protocol implemented by Gateway is documented primarily by Amazon AWS and specific integration topics are covered in the S3 Protocol Interface.

Multi-tenant framework

While Swarm already provides multi-tenant separation of content, Gateway builds upon that foundation by formally defining scopes within the storage system in order to provide developers with a proven framework for organizing and managing a cloud storage system. These are the scopes defined by Gateway:



Note that, while Caringo only defines the role of owner, you can create simple to sophisticated role based access control (RBAC) definitions as required for your organization using Gateway's access control policies. The Cluster, Tenant, Domain and Bucket "admins" shown in the diagram above are common roles that can be used but are not required or hard-coded into the system. For the purpose of explaining the scopes and their purposes, these roles will be assumed to be used in the system.

- Root Scope. The root scope exists on the Gateway servers' file systems as the configuration information necessary to bootstrap the cloud storage system. It contains the top-level definition of the identity management system and the overall access control policy for the entire cluster. The Gateway system administrations manage the resources at this level through standard Linux administration tools.
- Cluster Scope. The cluster scope is the top-level control point within the object storage system. The cluster administrations operating in this scope are the super users within the cloud storage system and have the ability to create and access all content within the system. Through the Content Portal or using management API calls, they create lower-level scopes, such as tenants and storage domains, and they can delegate management duties to those lower-level scopes to less privileged users.
- Tenant Scope. The tenant scope is a formalized concept that exists only within Gateway and not within Swarm. A tenant is a hierarchy that owns one or more storage domains. Each tenant scope can define their own identity management system so that the users and groups within them are separated from those in other tenants. The tenant administrators have the ability to create and access storage domains on behalf of the tenant and they can delegate management duties for the storage domains that they create. The tenant scope does not store end-user data; it is only a meta store for information about the tenant, its users, and its storage domains.
- Domain Scope. The domain scope is directly tied to a Swarm storage domain and is where end-user data is kept. The SCSP and S3 storage protocols create and use data within the domain scope. While the domain scope can inherit user and group identity information from its tenant, it also has the ability to define its own identity management system. The domain administrators can create and access all content within the storage domain. They can optionally delegate control of storage buckets to individual users or groups.
- Bucket Scope. The bucket scope is directly tied to a bucket that exists within the Swarm storage domain. While access control policies can be defined for every bucket, there is no option for an identity management system definition at the bucket scope. All buckets with a domain share the domain's identity management system

definition.

- IDM. Identity management system connection information is stored within IDSYS objects and they are the source of user and group information and the authentication system.
- Access Policy. The Policy objects contain the rules for access control to content within the system. This includes control of all operations through the Storage API and the Management API. Policies are associated with every scope within the storage system.

Management API

Separate from the Storage API for end-user content, the Gateway implements a storage management API as an integration point for cloud management platforms and developers that need to automate the provisioning and management of the cloud storage system.

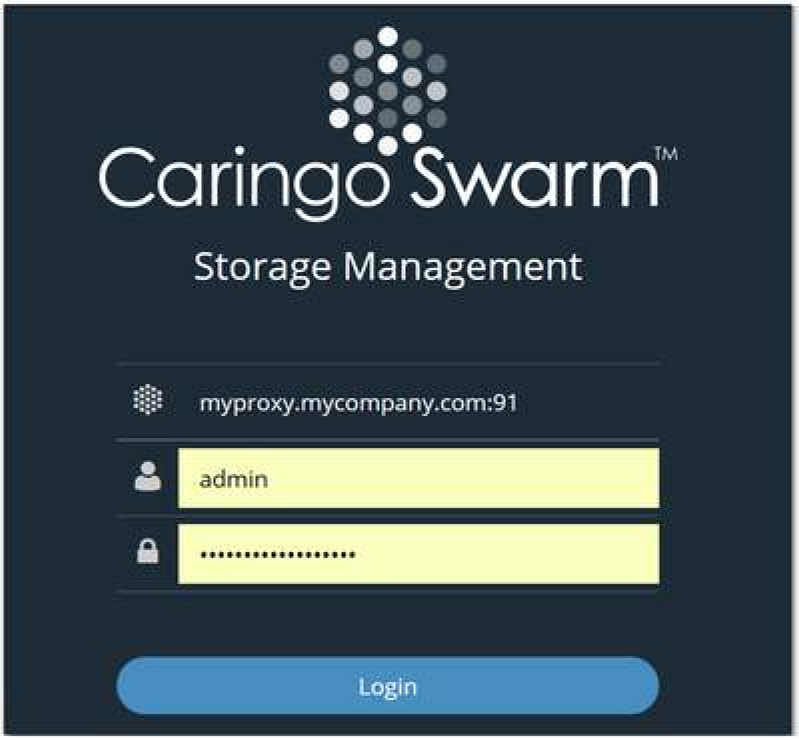
Service Proxy

Service Proxy is an optional front-end protocol for the Content Gateway that allows the Swarm storage UI to administer storage nodes on a back-end storage network. The Service Proxy is enabled and configured within the Gateway's configuration file. The Service Proxy provides access to the management API that is built into the Swarm cluster nodes, leveraging the same IDSYS authorization and authentication as your Content Gateway. With the Service Proxy, you can host the Storage Management UI and API from a server that is accessible to your administrators and have the Service Proxy manage communication with the Swarm cluster. Doing so gives you a single access point for managing and monitoring an entire Swarm cluster.

> Best practice
>
> Enable the Service Proxy for cluster admins only, to grant them alone access to the cluster's Storage Management UI and API, in addition to all content interfaces (Content Portal and the SCSP and S3 APIs). Disable Service Proxy for all other users (end users, tenants, customers), who are restricted to the content interfaces.

Using the Service Proxy

To enable users to log in to the Service Proxy, provide them with the new URI information to enter in the Swarm UI login screen:



Cluster URI	Rather than entering the IP address or hostname of a Swarm storage cluster node, use the Service Proxy hostname or IP instead. When using a hostname, ensure that DNS resolves the name to front-end IP address of the Content Gateway instance that is running Service Proxy. Required The Service Proxy port must be specified as well.
User Name	Specify a user and password that is known to the identity management system defined by the root IDSYS for the Content Gateway.
Password	

How the Service Proxy Works

The Service Proxy servlet listens on the specified port and handles two types of requests on the same port:

- Storage cluster management API requests, targeting storage nodes
- Elasticsearch query requests, targeting Elasticsearch nodes

Authentication and authorization for the Service Proxy uses Content Gateway's root IDSYS and root Policy.

> See Gateway Configuration for information on configuring the Service Proxy and the authentication/authorization.

Content Gateway Authentication

- Gateway Identity System
- Gateway Access Control Policies

Gateway Identity System

The Content Gateway identity system is implemented by the front-end Gateway component and allows for one or more user data store configurations. The user data store is responsible for authenticating users via a password and for defining group membership for users. Content Gateway currently supports LDAP and Linux PAM as the user data store. Microsoft Active Directory can be used through its LDAP interface or via PAM with a Kerberos configuration on the Gateway server.

The configuration of the identity system exists as IDSYS documents that are stored in the following locations:

- Root IDSYS file
- Tenant IDSYS sub-resource
- Storage domain IDSYS sub-resource

An IDSYS document contains the information necessary to connect with the identity system and defines the organization of users and groups within the identity system. While an IDSYS document may only define one back-end identity system, different back-end systems can be used for different tenants and storage domains within the cluster. For example: use PAM in the root IDSYS and use LDAP in the storage domains.

The root IDSYS is stored in this JSON file:

```
/etc/caringo/cloudgateway/idsys.json
```

This file must be kept synchronized between all Gateway servers. The `idsys` sub-resource for a tenant or storage domain is kept within the cluster, is shared among all Gateway servers, and is accessed through the Gateway Management API or the Storage API.

> Note
>
> The root IDSYS configuration file must exist and must contain a valid JSON string or be blank. The minimum valid JSON content is "{}".

- IDSYS Document Format

IDSYS Document Format

IDSYS documents are JSON-formatted objects and are specific to the back-end identity management system: LDAP or PAM.

- Common IDSYS Fields
- LDAP Identity System
- LDAP IDSYS Fields
- PAM Identity System
- Modifying IDSYS
- IDSYS Precedence Model
- Qualification of User and Group Names

Common IDSYS Fields

Below are the common fields within all IDSYS documents. Fields that are specific to the back-end identity management system are broken out into separate sections.

Field	Required	Description
name	No	Name of the IDSYS document; value is not used by Gateway
description	No	Description of the IDSYS document; value is not used by Gateway
comments	No	Comments about the IDSYS; may be any valid JSON object type
cookieName	No[1]	Cookie name used to store authentication token. Example: "token"
tokenPath	No[1]	URI path used for token authentication. Example: "/.TOKEN/"
tokenAdmin	No[1]	User that is allowed to view and delete authentication tokens for other users.

[1] For details regarding token-based authentication, see Token-Based Authentication.

When a user authenticates to the Gateway using HTTP Basic authentication (that is, not token-based authentication and not S3 HMAC), the user's password is stored in the normal field for LDAP or PAM and it may be hashed in whatever formats are supported by the system. For LDAP, this field is normally userPassword and, for PAM with the traditional Unix authentication mechanism, it is the second field in the `/etc/shadow` file.

LDAP Identity System

This is an example of an IDSYS document that uses LDAP. It contains fields that are specific to LDAP as well as fields that are common to all IDSYS documents.

```
{"ldap": {
 "name" : "idsys-ldap",
 "description": "LDAP identity management configuration",
 "protocol" : "ldaps",
 "ldaphost": ["ldap.example.com", "ldap-sec.example.com"],
 "ldapport": 636,
 "adminDN": "uid=YOURUSERNAME,ou=Users,dc=example,dc=com",
 "adminPassword": "YOURPASSWORD",
 "userBase": "ou=Users,dc=example,dc=com",
 "groupBase": "ou=Groups,dc=example,dc=com",
 "userFilter": "objectclass=account",
 "groupMemberUidAttr": "memberUid",
 "cookieName": "token",
 "tokenPath": "/.TOKEN/",
 "tokenAdmin": "superuser@admindomain.example.com"
} }
```

LDAP IDSYS Fields

These are the fields within the IDSYS document that are specific to the LDAP backend identity management system.

Field	Default	Required	Description
ldaphost		Yes	Host name or IP of the LDAP server as a string or a multiple servers as a list. Example: ["1.1.1.1", "1.1.1.2"]
ldapport		Yes	Port the LDAP service is running on
protocol	ldap	No	Set to "ldap" or "ldaps"
referrals	follow	No	Set to "follow" or "ignore" to control how referrals are handled
adminDN		Yes	DN used to bind to the LDAP server for queries
adminPassword		Yes	Password for adminDN user
userBase		Yes	DN where users are defined
groupBase		Yes	DN where groups are defined
uidAttribute	uid	No	Attribute name containing user's ID. Examples: "uid" for OpenLDAP and ApacheDS; "sAMAccountName" for Active Directory
userFilter		Yes	Filter for user objects. Example:"objectclass=account"

groupMemberUidAttr		Yes[1]	Group attribute whose values contain uid of member. Example: "memberUid" if OpenLDAP is configured for groups with "objectclass=posixgroup"
groupMemberDNAttr		Yes[1]	Group attribute whose values contain DN of member. Example: "member" if OpenLDAP is configured for groups with "objectclass=groupOfNames"; also common with Active Directory
s3SecretKeyAttr		No[2]	**Deprecated** User attribute whose value contains the user's S3 secret key in plain-text. If "userPassword" is used, you must ensure that it has a plaintext value since this is not the normal handling of this attribute.

[1] The `groupMemberUidAttr` and `groupMemberDNAttr` parameters are mutually exclusive and you must only define one of them in IDSYS.

[2] The `s3SecretKeyAttr` parameter is only needed when using S3 Protocol Personality with a user password stored in LDAP. It is not required when using token authentication exclusively.

The `adminDN` and `adminPassword` parameters define the credentials with which the Gateway binds to the LDAP system in order to perform queries and read records for users and groups. The `adminDN` entity within LDAP needs to have read level access (`rscdx` privileges) within the LDAP tree. It is not necessary to grant write or manage level access to Gateway.

- A user's name in an access control Policy document is the value of the LDAP attribute named by the `uidAttribute` parameter. By default this will be the `uid` attribute of a user's LDAP record.
- A group's name in an access control Policy document is the `cn` attribute for the group LDAP entity. The name of this attribute cannot be configured. A group's name may contain spaces and other non-alphanumeric characters.

Important

If you are using LDAPs with self-signed certificates, you must add your signer's public key PEM file to your Java keystore to avoid a SunCertPathBuilderException when Gateway queries the LDAP server. Although previously this required running java keytool, you should now use the CentOS/RHEL utility update-ca-trust to add any CA to your system.

```
# cp ca.pem /etc/pki/ca-trust/source/anchors/
# update-ca-trust extract
```

Important

Nested/recursive groups, such as the built-in groups in Active Directory, are not supported by Gateway.

PAM Identity System

There are no fields within the IDSYS document that are specific to the PAM back-end identity management system. If you are using PAM, follow this process to implement your identity management:

1. Because the `root` user (uid=0) on this Content Gateway server cannot be used to authenticate to the Gateway, create another user (such as `superuser@admindomain.example.com`) on this server for this purpose.
2. Copy and paste this example into your IDSYS document: `/etc/caringo/cloudscaler/idsys.json`.

```
{"pam": {
    "name" : "idsys-pam",
    "description": "PAM identity management configuration",
    "cookieName": "token",
    "tokenPath": "/.TOKEN/",
    "tokenAdmin": "superuser@admindomain.example.com"
} }
```

3. Update the `tokenAdmin` to match your authentication user.

Modifying IDSYS

The root IDSYS configuration is stored in the idsys.json file on the Gateway server's disk so that it is always available and so that an administrator can always modify it. In other words, you can't lock yourself out of the storage cluster. Changes to the local file take effect without the need to restart the Gateway.

> Important
>
> When more than one Gateway server is deployed, it is crucial that the root IDSYS document is synchronized across all servers.

> See Defined ETC Documents for modifying a tenant or storage domain's sub-resource through the management API.
>
> See SCSP Context Sub-resources for details on modifying a storage domain's sub-resource through the storage API.

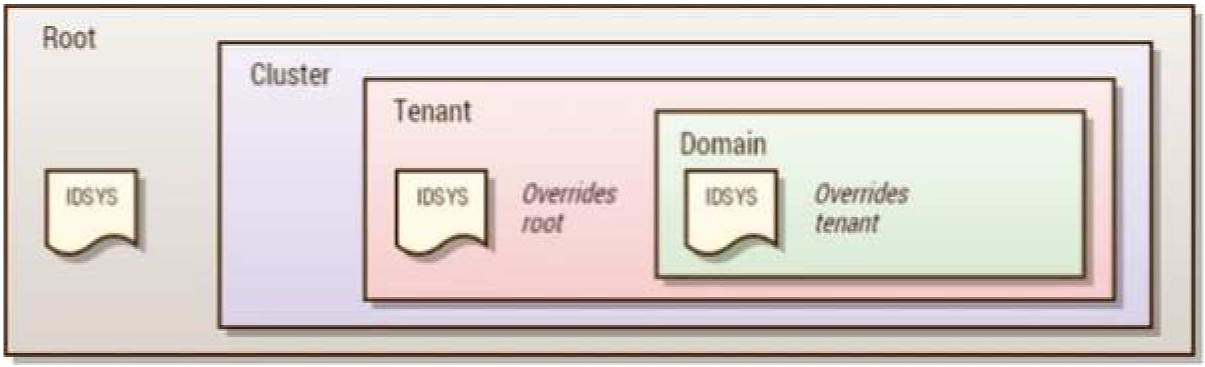
When updating an IDSYS sub-resource through the management API or the storage API, the entire JSON document with all fields must be provided in the update request, even if only one field is being modified.

> Caution
>
> Be sure to protect permission to read and update the IDSYS document for a domain from untrusted users. In deployments where a service provider allows customers to manage content within domains, the service provider will normally maintain sole privilege to access the IDSYS document. This includes retaining ownership of the domain objects.

IDSYS Precedence Model

The identity system is described by IDSYS documents can exist at the root, tenant, and storage domain within the system. When IDSYS documents exist at multiple levels in the hierarchy, the lowest level overrides the higher levels.

When a lower level lacks an IDSYS, it inherits from a higher level.



For example, if only a Root IDSYS exists, all tenants and all storage domains will inherit from the Root IDSYS. In this case, there will only be one identity management system with one set of users and groups. If, however, the tenants each defined their own IDSYS, each tenant and the storage domains owned by them would share their own identity system that would be separate from the Root IDSYS. In this second case, the storage domains would inherit from their Tenant IDSYS.

The IDSYS inheritance also works at the field level, meaning that tenant and storage domain IDSYS documents can choose to override specific fields. For example, if tokenAdmin is defined in the Root IDSYS and not in the tenant or domain IDSYS, its value will be inherited by the tenant and domain levels. Similarly, the Root IDSYS may define the LDAP adminDN and adminPassword and let the tenant and domain IDSYS documents override the userBase and group Base values.

- Single Company
  - In this scenario, the company has one identity management system, there is one tenant per business unit, and each business unit has one or more storage domains. This scenario is likely with a private cloud that serves a single company. The configuration in this scenario is that the Root IDSYS defines the configuration of the identity management system and there are no IDSYS definitions for the tenants and storage domains. Therefore, the tenants and storage domains will inherit from the Root IDSYS using a single source of users and groups.
- Service Provider / Distributed Company
  - In this scenario, a storage MSP, or a large company that has business units each with their own identity management systems, has multiple user/group sources. The configuration in this scenario is that the Root IDSYS defines the cluster administrator users and groups and the Tenant IDSYS documents define the users and groups for each customer or business unit. The storage domains do not define an IDSYS so that they inherit the definition from the tenant and share the users and groups with the other storage domains owned by their tenant.
- Service Provider with Resellers
  - This is an extension of the previous scenario except that each tenant could be a reseller offering storage domains to separate, unrelated companies. In this case, each storage domain would define an IDSYS that would override the Tenant IDSYS allowing a different set of users and groups for each storage domain. This scenario is not mutually exclusive with the previous one: a hybrid of the two is possible where some domains override the IDSYS of their tenant, and others do not.

Qualification of User and Group Names

Qualification of user and group principal names refers to explicitly denoting the tenant name or storage domain from where the principal comes when referred to in access control policies and `x-owner-meta` headers.

user group	non-qualified	Principal from the same IDSYS scope as the content being accessed
user@domain group@domain	fully-qualified	Principal from a specific storage domain's IDSYS scope
user+tenant group+tenant	fully-qualified	Principal from a specific tenant's IDSYS scope
user@ group@	fully-qualified	Principal from root scope only

Normally, Gateway does not require qualification of user and group names in Policy documents when the users and groups are resolved against the same IDSYS source that the resource uses. For example, within a storage domain that has an IDSYS, objects owned by users authenticated by the domain's IDSYS and access control policies written to refer to users and groups from the domain's IDSYS do not need to include an `@domain` or `+tenant` suffix on principal names.

When a user creates an object in a storage domain, qualification of the principal as `user@domain` or `user+tenant` occurs automatically with the default assignment of the `x-owner-meta` header value if the user authenticated from an IDSYS that is different than the one used by the storage domain. Applications can also assign ownership to objects in a storage domain that uses one IDSYS to a user from another domain that uses a different IDSYS.

Qualification of user and group principals is done in Policy documents when it is necessary to refer to users and groups from an IDSYS that is different for the IDSYS in use by the tenant or storage domain in which these users or groups are accessing content.

> See Policy Document.

Best practice
Token administrators, defined in an IDSYS document, should be fully qualified. Since token administrator is a privileged permission, this is a good practice in order to avoid ambiguity in a situation where a storage domain may inherit its IDSYS from the tenant or root scope.

Gateway Access Control Policies

The Content Gateway provides a rich access control mechanism that allows for coarse to fine-grained control over user access to content within a storage domain and administrative actions within the management API. Access control is defined within Policy documents that may specify permissions on specific objects when necessary. Access control Policy documents are stored in the following locations:

- Root Policy file
- Tenant Policy sub-resource
- Storage domain Policy sub-resource
- Bucket Policy sub-resource

The root Policy document is stored in a JSON file:

```
/etc/caringo/cloudgateway/policy.json
```

This file must be kept synchronized between all Gateway servers. Changes to the local file take effect without the need to restart the Gateway. The policy sub-resource for tenants, storage domains, and buckets is kept within the cluster, is shared among all Gateway servers, and is accessed through the management API or storage API.

> Note
>
> The root Policy configuration file must exist and must contain a valid JSON string or be blank. The minimum valid JSON content is "{}".

- Policy Document
- Policy Evaluation
- Modifying Policies

Policy Document

- Policy Document Fields
- Policy Format
- Principals
- Request Actions
- Policy Conditions

> Best practice
>
> To create a policy for domain administrators to have broad permissions, you can list each Action that they are allowed to perform. However, it is easier and less error-prone to list the few actions that they cannot perform (such as DeleteDomain and CopyDomain), by replacing the entire Action statement with the NotAction statement:

```
{
   "Version":"2016-10-17",
   "Statement":[
      {
         "Sid":"Grant all except excluded domain operations to
admins2",
         "Resource":"/*",
         "Effect":"Allow",
         "Principal":{
            "group":[
               "admins2"
            ]
         },
         "NotAction":[
            "CopyDomain",
            "DeleteDomain"
         ]
      }
   ]
}
```

Policy Document Fields

Policy documents are JSON-formatted objects.

Field	Sub-field	Description
Version		optional; if used must be "2012-10-17" or "2008-10-17" to match S3
Id		optional name to describe the policy
Statement		a list of rules
	Sid	unique identifier for each statement
	Effect	whether the rule allows or denies an operation
	Principals	defines users and groups to which the rule applies
	Action	content operation (request)

	NotAction	logical inverse of Action; supports all Action values
	Resource	path for which the rule applies; "/" prefix is optional
	Condition	conditional requirements for the request

Policy example

```
{
  "Version": "2008-10-17",
  "Id": "Comics-and-Superheros",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "1",
      "Principal": {
        "group": [
          "Development",
          "QA",
          "testers@partner.net"
        ],
        "user": [
          "jdoe",
          "gcarlin@funny.com",
          "tony+starkindustries"
        ]
      },
      "Action": [
        "GetObject"
      ],
      "Resource": "/*",
      "Condition": {
        "StringLike": {
          "Referer": [
            "*example.com"
          ]
        }
      }
    }
  ]
}
```

Policy Format

The format of Gateway's Policy document is modeled after Amazon S3 bucket policies. There is only one policy document per tenant, per storage domain, and per bucket. In other words, updating a policy document completely replaces any existing document. When evaluating a request, the root Policy plus the relevant tenant, storage domain, and bucket policies are all first merged together.

The policy is a set of rules specifying the conditions under which a request is allowed or denied within the storage API or the management API (such as create a domain, bucket, or object). Each statement in a policy specifies the principals allowed or denied the ability to perform an action on a resource. These are examples of each of these elements of a policy specification.

- Principals: user "jdoe", group "Development"
- Action: GetObject, DeleteObject
- NotAction: CopyDomain, DeleteDomain
- Resource: "mybucket/private/*", "mybucket/photos/picture01.jpg"
- Condition: "StringLike" : { "Referer" : ["*example.com"]} }
- Effect: "Allow", "Deny"

> Best practice
>
> For best performance, make your Policy documents as small as reasonable. Performance is affected because the statements have to be evaluated for each request to check if the resource matches. Choose the most efficient definitions; for example, NotAction lists may be shorter than Action lists.

The `Id` field is provided for end-user use only and is ignored by the policy evaluation. The `Version` field is optional and, if defined, the only valid value is "2008-10-17".

> Caution
>
> Do not prefix resources with "`arn:aws:s3:::`" and actions with "`s3:`" strings: these are ignored by the policy evaluation and hurt performance.

Principals

Principals are users or groups to whom the access control statement applies. This includes the specification of an anonymous user that has not authenticated. The format is:

Specifying authenticated users

```
"{user|group}":[{list- of- principals}]
```

These notation formats are recognized for anonymous principals:

Anonymous principals

```
Principal: "*"
Principal: { "anonymous" : ["*"] }
Principal: { "anonymous" : "*" }
Principal: { "AWS" : ["*"] }
Principal: { "AWS" : "*" }
```

This is an example that includes users and groups from other storage domains and tenants.

Including users and groups

```
"Principal" : {
    "user":["vcerf", "timbl", "ltorvalds@kernel.org"],
    "group":["Development", "QA+acme", "Development+acme"],
}
```

> See Cross-tenant Access Control below for the meaning of the `@domain` and `+tenant` suffixes for users and groups.

The following table explains the meaning of the different forms of the principal specifications.

Principal	Description
"anonymous":["*"]	An anonymous, unauthenticated user
"user":["*"]	Any authenticated user from any IDSYS scope
"user":["*@austin"]	Any authenticated user from the 'austin' storage domain's IDSYS (or its inherited IDSYS if applicable)
"user":["*+texas"]	Any authenticated user from the 'texas' tenant's IDSYS (or its inherited IDSYS if applicable)
"user":["gcarlin"]	A user named 'gcarlin' from this scope's IDSYS (or inherited IDSYS if applicable). This is a non-qualified user name since no domain, tenant, or root scope is specified.
"user":["gcarlin@cars"]	A user named 'gcarlin' from the 'cars' storage domain's IDSYS (or its inherited IDSYS if applicable)
"user":["gcarlin+movies"]	A user named 'gcarlin' from the 'movies' tenant's IDSYS (or its inherited IDSYS if applicable)

"user":["gcarlin@"]	A user named 'gcarlin' only from the root IDSYS
"group":["admins"]	Any member of the group named 'admins' from this scope's IDSYS (or inherited IDSYS if applicable). This is a non-qualified group name since no domain, tenant, or root scope is specified.
"group":["admins@hockey"]	Any member of the group named 'admins' from the 'baseball' storage domain's IDSYS (or its inherited IDSYS if applicable)
"group":["admins+sports"]	Any member of the group named 'admins' from the 'sports' tenant's IDSYS (or its inherited IDSYS if applicable)
"group":["admins@"]	Any member of the group named 'admins' only from the root IDSYS

In many cases, it is unnecessary to explicitly grant permissions to the user named by the `X-Owner-Meta` header, of a tenant, storage domain, or bucket because they are granted permission for all operations by default. This default permission means that careful consideration should be given when assigning ownership to another user — especially for tenants or storage domains.

> Linux root User
> The `root` user (uid=0) is not allowed to authenticate to Gateway using Linux PAM.

User and group principals can be written as fully-qualified or non-qualified within Policy documents. A fully-qualified principal is one that defines the domain, tenant, or root scope (example: "gcarlin@cars"). A non-qualified principal does not include the scope for the principal (example: "gcarlin"). The scope of a principal defined the starting point for looking for the first available IDSYS definition with which to look for the principal. If the principal is non-qualified, the starting point for the IDSYS search starts from the point of the resource that is being requested.

The IDSYS precedence model allows the root scope to be overridden by the tenant and the domain scopes. The search order when looking for the first available IDSYS is to search backwards from domain to tenant to root. If a group is specified as "admins@hockey" for requests within the "hockey" storage domain, the Gateway look for the first IDSYS that is defined at the domain, then the tenant, then at the root. Upon finding the first IDSYS definition, the group "admins" from only that IDSYS will be used.

Request Actions

Policy documents are used to control actions with both the Management API and the Storage API. The policy actions that apply to a given request depends upon the API being used, management or storage. The list of Policy documents that are merged together for policy evaluation is determined by the path in the API hierarchy being accessed.

When request actions are listed within an Action field, these actions are included in the Effect field result. For example, if the effect is "Deny", then the listed actions would be explicitly denied. When request actions are listed within a `NotAction` field, these actions are excluded from the Effect field results. For example, if the effect is "Deny", then the all actions other than those listed would be explicitly denied.

Policy Actions for Administration (Management API)

These are the Management API actions that can be controlled within a Policy document and the applicable scope where the actions are used. If an action is granted or denied within a scope where it is not applicable, it has no effect.

Scope indicates where the policy action is applicable: (R) root, (T) tenant, (D) domain, (B) bucket.

Manage	Action	Scope	Description
Global	*	R,T,D,B	all actions
Tenants	ListTenants	R	List all tenants
	CreateTenant	R	Create a new or change an existing tenant
	GetTenant	R,T	Retrieve tenant properties
	DeleteTenant	R,T	Permanently remove tenant properties
	ListEtc	R,T	List documents associated with a tenant
Domains	ListDomains	R,T	List the domains owned by the _system tenant
	CreateDomain	R,T	Create a domain for the _system tenant
	GetDomain	R,T,D	GET a domain
	DeleteDomain	R,T,D	Delete a domain
Policies	ListEtc	R,T,D	List documents associated with a tenant or a storage domain
	PutPolicy	R,T,D	Create or update an access control policy JSON document
	GetPolicy	R,T,D	Read an access control policy JSON document
	DeletePolicy	R,T,D	Permanently remove an access control policy JSON document
Authentication Tokens	TokenAdmin	R,T,D	Create and list authorization tokens for other users in the same scope
	CreateToken	R,T,D	Create an authentication token
	ListTokens	R,T,D	List user authentication tokens
	ValidateToken	R,T,D	Read an authentication token
	DeleteToken	R,T,D	Delete an authentication token

Policy Actions for Content (Storage API)

This table defines the Storage SCSP API actions that can be controlled within a Policy document and the applicable scope where the actions are used. If an action is granted or denied within a scope where it is not applicable, it has no effect.

> For Amazon S3 operation permissions, see Specifying Permissions in a Policy in the AWS documentation.

Scope indicates where the policy action is applicable: (R) root, (T) tenant, (D) domain, (B) bucket.

Storage	Action	Scope	Description
Global	*	R,T,D,B	all actions
Quotas	PutQuota	R,T,D,B	Create or update quota configuration settings on a tenant, domain, or bucket. This is not granted by "*"; it must be granted explicitly.
	GetQuota	R,T,D,B	Retrieve quota configuration settings on a tenant, domain, or bucket. This is not granted by "*"; it must be granted explicitly.
Objects	GetObject	R,T,D,B	Retrieve an object and its metadata
	GetObjectAcl	R,T,D,B	Retrieve the Access Control List (ACL) settings on an object
	CreateObject	R,T,D,B	Add a new object and metadata to a domain (unnamed) or bucket (named). There is no S3 equivalent, and it is permissioned separately here because Swarm distinguishes POST and PUT.
	AppendObject	R,T,D,B	Append to an object
	CopyObject	R,T,D,B	Update metadata on an object while keeping the same object name
	PutObject	R,T,D,B	Update an object PutObject, unlike S3, is for overwriting an existing object, not creating new one (CreateObject).
	PutObjectAcl	R,T,D,B	Set an object's access control list
	DeleteObject	R,T,D,B	Permanently remove an object and its metadata
Buckets	CreateBucket	R,T,D	Create a bucket
	PutBucket	R,T,D	Synonym for CreateBucket
	PutBucketAcl	R,T,D	Set bucket access control list
	PutBucketCORS	R,T,D,B	Set CORS configuration on a bucket
	GetBucket	R,T,D,B	GET or HEAD a bucket
	GetBucketAcl	R,T,D,B	Get bucket access control list
	GetBucketCORS	R,T,D,B	Get CORS configuration on a bucket
	CopyBucket	R,T,D,B	Update the metadata on a bucket while keeping the same bucket name

	DeleteBucket	R,T,D,B	Delete a bucket
	ListBucket	R,T,D,B	List and search the objects contained in a bucket
Bucket Sub-Resources	PutPolicy	R,T,D,B	Create or update a policy, xform, sub-resource for a bucket. While an owner of bucket can always overwrite the policy, this allows a group to be granted this permission.
	GetPolicy	R,T,D,B	GET or HEAD the policy, xform, sub-resource for a bucket
	DeletePolicy	R,T,D,B	Permanently remove a policy, xform, sub-resource for a bucket
Domains	CreateDomain	R *	Create a domain
	GetDomain	R,T,D	GET or HEAD a domain
	CopyDomain	R,T,D	Update the metadata on a domain while keeping the same domain name
	DeleteDomain	R,T,D	Permanently remove a domain
	ListDomain	R,T,D	List and search all objects (buckets and unnamed objects) within the domain
	ListDomains	R *	List all domains that exist in the cluster
Domain Sub-Resources	PutPolicy	R,T,D	Create or update a policy, xform, idsys, sub-resource for a domain
	GetPolicy	R,T,D	GET or HEAD policy, xform, idsys, sub-resource for a domain
	DeletePolicy	R,T,D	Permanently remove a policy, xform, idsys, sub-resource from a domain
Authentication Tokens	CreateToken	R,T,D	Add a new authentication token
	ListTokens	R,T,D	Search for and list authentication tokens
	ValidateToken	R,T,D	Read and verify an authentication token
	DeleteToken	R,T,D	Permanently remove an authentication token

* The Tenant scope does not apply to SCSP CreateDomain and ListDomains because these operations refer to the _ system tenant implicitly: there is no "tenant" access through SCSP.

Policy Conditions

The Condition field of the Policy document allows for conditions other than user and action to be placed upon the request.

Conditionals in a Policy document allow for the value matching functions:

```
StringEqualsIgnoreCase
StringLike
```

- `StringEqualsIgnoreCase` matching is an exact match ignoring letter case.
- `StringLike` matching allows for glob-style pattern matching for the values and ignores letter case.

All matching conditionals are evaluated on each request similar to the way paths are matched to resources.

Referral Condition

Gateway supports conditional requirements on the value of the HTTP/1.1 `Referer` header.

> Note
> In order to match the header specification in RFC 7231 5.5.2, Gateway uses the same spelling (or misspelling) of "referer."

The `Referer` header is commonly provided by web browsers when retrieving content that was referenced by an HTML document. While the browser, or HTTP client library, is free to provide any value they wish for this header, it is commonly used to detect the source reference for a resource request. The following logical evaluation takes place for the referral matching condition:

- If there is no `Referer` header in the request, then only policy statements without `Referer` conditions will be considered for authorization.
- If there is a `Referer` header in the request, policy statements without `Referer` conditions will be considered for authorization.
- If there is a `Referer` header in the request and there are policy statements with `Referer` conditions, then the policy statement will be considered for authorization if the values match. Values are matched as follows:
    - If there are multiple values in a conditional value list, the condition matches if any value matches for the condition-key. It is a logical OR match.
    - If there are multiple condition-types in a condition-block, all conditions must match. It is a logical AND match.

Rules that match a request are chosen from available rules according to Resource, Action, referral Conditions as described. Among those rules, further filtering is done by Principal and lastly according to deny/apply semantics. Note that referral conditions are processed in all actions.

This example shows a conditional restriction that the `Referer` value must be `example.com` or `another.com.`

```
{
  "Statement" : [
    {
      "Condition" : {
        "StringEqualsIgnoreCase" : {
          "referer" : ["example.com","another.com"],
          ...
          }
      }
    }
  ]
}
```

This example shows a conditional restriction that uses the `StringLike` match in order to restrict the `Referer` value to any subdomain within a parent domain.

```
{
  "Statement" : [
    {
      "Condition" : {
        "StringLike" : {
          "referer" : ["*example.com"],
          ...
          }
      }
    }
  ]
}
```

While `Referer` header restrictions are commonly used to prevent bandwidth stealing due to cross-linking by unaffiliated web sites, the requesting client is being trusted to accurately populate this header. So, while in most cases, `Referer` header restrictions work well enough to prevent unauthorized cross-linking, this is not a mechanism that should be relied upon to provide site security.

> Note
> The referral conditions only applies to the Storage API and not to the Management API operations.

Prefix Condition

Gateway supports conditional requirements on the prefix match. This can be used in order to require that object searching be constrained to a set of prefix patterns.

This is an example of a policy that only allows the listing of objects if you specify a prefix query argument.

```
{
   "Sid":"AllowListingOfSharedFolder",
   "Action":[
      "ListBucket"
   ],
   "Effect":"Allow",
   "Resource":[
      "usersbucket"
   ],
   "Condition":{
      "StringLike":{
         "prefix":[
            "home/Shared/*"
         ]
      }
   }
}

Allowed: GET /usersbucket?format=json&prefix=home/Shared/
Denied: GET /usersbucket?format=json&prefix=home/Shared/
```

Policy Evaluation

- Administrator Roles
- Object Ownership
- Evaluation Precedence

These are the administrator roles and the rules for the order of policy evaluation. Administrator roles are based on ownership and access permissions defined within the Gateway and storage contexts. While these are all just users, possibly in different LDAP DNs and/or PAM, and could all be the same set of users, the following definitions are useful for describing the normal responsibilities that each classification of administrator has. They are normal boundaries where access control is segregated.

> Role-based Access Control
>
> While Caringo only defines the role of owner, you can create simple to sophisticated role-based access control (RBAC) definitions as required for your organization using the Gateway's access control policies. The Cluster, Tenant, Domain, and Bucket "administrators" are common roles that can be used, but they are not required or hard-coded into the system.

Administrator Roles

<table>
<tr><th>Type</th><th>Administrator Role</th><th>Example</th></tr>
<tr><td>Cluster</td><td>Define who can create tenants and non-tenanted domains
Analogous to the Swarm administrator except that they are defined in an external identity management system. This user or group is specified in the policy.json root Policy configuration file.</td><td>This policy.json file defines and grants full permissions to the cluster administrators group called ClusterAdmins. The members of the ClusterAdmins group are the users that are cluster administrators and can often be the same people that maintain the physical infrastructure.
<pre>
{
  "Version": "2008-10-17",
  "Id": "ClusterAdminsPolicy",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "group": [
          "ClusterAdmins"
        ]
      },
      "Action": [
        "*"
      ],
      "Resource": "/*"
    }
  ]
}
</pre></td></tr>
</table>

<table>
<tr>
<td>Tenant</td>
<td>Define who can create domains for the tenant
Owner of the tenant object as specified by the X-Owner-Meta meta data header. It is common for the tenant administrator to create a Policy document for the tenant that grants permissions for a group of users to act on the same authority of the tenant administrator.</td>
<td>This tenant Policy document grants full access to a group called TenantAdmins whose members come from users within the acme tenant.
<pre>
PUT
/_admin/manage/tenants/acme/etc/policy.json
{
  "Version": "2008-10-17",
  "Id": "TenantAdminsPolicy",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "1",
      "Principal": {
        "group": [
          "TenantAdmins"
        ]
      },
      "Action": [
        ""
      ],
      "Resource": "/"
    }
  ]
}
</pre></td>
</tr>
</table>

<table>
<tr>
<td>Domain</td>
<td>Define who can create buckets and unnamed objects
Owner of the storage domain as specified by the X-OwnerMeta metadata header. It is common for the domain administrator, owner of the storage domain, to create a Policy document for the domain that grants permissions for a group of users to act on the same authority of the domain administrator.</td>
<td>This domain Policy document grants full access to a group called DomainAdmins whose members come from users within the domain.
<pre>
PUT http://DOMAIN/?policy
{
  "Version": "2008-10-17",
  "Id": "DomainAdminsPolicy",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "1",
      "Principal": {
        "group": [
          "DomainAdmins"
        ]
      },
      "Action": [
        ""
      ],
      "Resource": "/"
    }
  ]
}
</pre></td>
</tr>
<tr>
<td>Bucket</td>
<td>Define who can create named objects within the bucket
Owner of the bucket as specified by the X-Owner-Meta header. The bucket administrator, owner of the bucket, can attach a Policy document to the bucket that defines the access control policy for the bucket and its contents.</td>
<td>This bucket policy gives any authenticated user full access under `http://DOMAIN/mybucket/incoming/*` and gives users in the Finance group full access under `http://DOMAIN/mybucket/reports/*` .</td>
</tr>
</table>

```
PUT http://DOMAIN/mybucket?policy
{
  "Version": "2008-10-17",
  "Id": "MyBucketPolicy",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "user": [
          "*"
        ]
      },
      "Action": [
        "*"
      ],
      "Resource": "/mybucket/incoming"
    },
    {
      "Sid": "2",
      "Effect": "Allow",
      "Principal": {
        "group": [
          "Finance"
        ]
      },
      "Action": [
        "*"
      ],
      "Resource": "/mybucket/reports"
    }
  ]
}
```

Here, all of the objects are contained with the bucket context mybucket . The access control policy matches named objects with the prefixes incoming/ and reports/ within that bucket.

> Note
> Notice that the bucket name is included when specifying resources in the bucket policy.

Object Ownership

All objects created and updated through the Gateway will have the X-Last-ModifiedBy-Meta metadata value set to the authenticated user performing the request. If an object is created by an anonymous useR,The value will be blank. Additionally, unless the request includes an X-Owner-Meta metadata value, it will be assigned the value of the authenticated user or blank for anonymous. If X-Owner-Meta is blank, everyone will be considered to be the owner for policy evaluation.

- For users authenticated within the same domain as the objects, the format of the metadata values is simply the base user name. For example: "john".
- For users authenticated from a different domain than the objects, the format will be fully qualified with the authenticating domain. For example: "john@another.com".
- For users authenticated from the root IDSYS, the format will be user + "@". For example: "john@".
- Similarly, if an object is created by a cluster administrator using an administrative override, the format will be the same since the user is authenticated from the root IDSYS. For example "admin@".

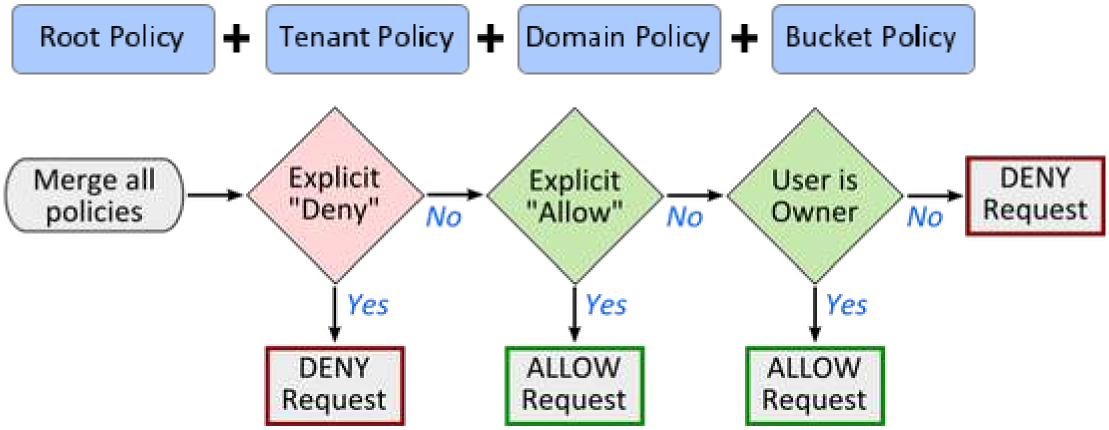
> See Modifying Policies.

Evaluation Precedence

When evaluating a user's authorization to perform a requested action, all relevant Policy documents are first merged together. Policies can exist at the root, tenant, storage domain, and bucket levels and, depending upon the request's level within the API hierarchy, one or more of these policies will be merged together for evaluation.

For example, if a user requests to read an unnamed object within a storage domain, the Root Policy, Tenant Policy, and Domain Policy will be merged together. Requesting the read of a named object within a bucket would also merge the Bucket Policy along with the root, tenant, and storage domain policies.

This is the policy evaluation logic:



The evaluation logic follows the precedence rules:

1. By default, all requests to a resource are denied to anyone except the owner specified by the header X-Owner-Meta on the resource.
2. An allow overrides any default denial from #1.
3. An explicit deny overrides any allows.

Note

The order in which the policies are evaluated has no effect.

The result of an explicit deny overriding allow is that if you deny an action to all users (`"user" : ["*"]`), even if you also explicitly allow yourself the same action, you will be denied the ability to perform that action.

If a request is for a non-existent domain or bucket context object, policy evaluation is short-circuited and an HTTP 404 response code is returned to the client. When the domain or bucket exists, policy evaluation returns an HTTP 403 if the user is not authorized to perform the action. The practical implication is that a user that is authorized to access a storage domain may be able to detect the existence of a bucket for which they do not have access. This paragraph only applies to context objects and not to objects that hold user content.

If the X-Owner-Meta header is blank or missing from a storage domain or bucket object, its ownership is anonymous and all users will match as the owner. The result is that everyone has owner permissions on the storage domain or bucket. It is a best practice to always assign ownership to context objects.

Modifying Policies

- Administrative Override
- Domain IDSYS and Bucket Policy Overrides
- User Name Login Formats
- Cross-Tenant Access Control
- Swarm Node Status Page Visibility

The root Policy configuration is stored in the policy.json file on the Gateway server's disk so that it is always available and so that an administrator can always modify it. Combined with an administrative login, this means that the cluster administrator cannot be locked out of the storage cluster.

Important

When more than one Gateway server is deployed, it is crucial that the root Policy document is synchronized across all servers.

> See Defined ETC Documents for modifying a Policy for a tenant, storage domain, or bucket through the management API.
>
> See SCSP Context Sub-resources for modifying the policy sub-resource for a domain or bucket through storage API.

Administrative Override

The Gateway's access control mechanism has a provision to allow cluster administrators to log into a tenant or storage domain and bypass all restrictions that may be contained within that tenant's or domain's policy sub-resource. This can be used to access content where the owner mistakenly cut-off all access and locked out their users. Another example is if the IDSYS for the storage domain has incorrect information within it that prevents all user logins.

The cluster administrator performs an administrative bypass request by putting a "!" before and an "@" after their username. The form of the user name portion of the login is: "!" + name + "@"

This instructs Gateway to use only the root IDSYS and root Policy documents when looking up the user and when evaluating whether an action is allowed. Any idsys or policy sub-resource associated with the tenant, storage domain, or buckets are ignored.

This example logs in as the user named admin and blanks the Policy document on the marketing.example.com domain so that only the owner who is identified by the X-OwnerMeta header of the domain object is allowed to operate in the domain.

```
curl -u '!admin@:adminpw' -X PUT
 -H 'Content-type: application/json'
 -d '' 'http://DOMAIN/?policy'
```

Presumably, after unlocking the storage domain, the owner would then re-submit a corrected Policy document that would allow the proper user access for the domain's content.

Domain IDSYS and Bucket Policy Overrides

Along the lines of the administrative override, Gateway provides a mechanism for accessing a storage domain by bypassing either the domain's IDSYS or a bucket's Policy.

In order to bypass the storage domain's IDSYS in favor of the root IDSYS, the user name for the request uses the form: user + "@". For example "psmith@". Requests performed using user names in the form "user@" are still subject to the domain and bucket Policy. Logins in this form only affect the authentication source for users.

In order to bypass a bucket's Policy, the user name for the request uses the form: "!" + user, such as "`!psmith`" or "`!psmith@other.com`". Requests of this form are authenticated using a domain IDSYS and are still subject to the domain Policy. This form of login can be used by the domain administrator to modify a the bucket Policy for another user's bucket. Notice that this override also works when the domain owner is from another tenant domain.

User Name Login Formats

These are the meanings for the different user name formats used to authenticate with Gateway.

User Name	Effect	Restrictions
user	Use this domain's IDSYS and domain + bucket Policy sub-resource.	User must be able to log into this domain's IDSYS.
user@otherdomain	Use other domain's IDSYS and this domain + bucket Policy sub-resource.	Other domain must exist in same storage cluster as this domain. User must be able to log into other domain's IDSYS.
user+othertenant	Use other tenant's IDSYS and this domain + bucket Policy sub-resource	Other tenant must exist in same storage cluster as this domain. User must be able to log into other tenant's IDSYS.
!user	Use this domain's IDSYS and domain Policy; ignore any bucket Policy.	Only domain owner can use this.
!user@otherdomain	Use other domain's IDSYS and this domain's Policy; ignore any bucket Policy.	Only domain owner can use this; owner is from another domain.
!user+othertenant	Use other tenants's IDSYS and this domain's Policy; ignore any bucket Policy.	Only domain owner can use this; owner is from another tenant.

user@	Use root IDSYS (ignore domain IDSYS) and use domain + bucket Policy sub-resource.	User must be able to log into root IDSYS.
!user@	Use root IDSYS (ignore domain IDSYS) and root Policy (ignore domain + bucket Policy sub-resource).	User must be able to log into root IDSYS.

Cross-Tenant Access Control

The access control policy evaluation in Gateway allows for the specification of users and groups that exist in other tenants and other storage domains within the Swarm cluster. This is done by appending `@domain` or `+tenant` qualifications onto a user or group name.

> See "Qualification of User/Group Names" in IDSYS Document Format for a full explanation for user and group qualification.

While there is no limit on the number of other domains that may be specified in a Policy document, all of the tenants and storage domains must exist within the same storage cluster.

Applications should not use `@domain` and `+tenant` qualifications unless the users and groups to which the principals refer are actually outside of the storage domain or tenant in which the policy resides. When users and groups are fully qualified, an IDSYS must exist within the referenced tenant or storage domain.

Swarm Node Status Page Visibility

The Swarm node status page of the legacy Admin Console is presented in response to a "GET /" request. If access to the resource "/*" is granted to anonymous or any user, the result will be that they will be authorized to request "/" and will be able to view the node status page for the back-end storage cluster. Note that due to connection pooling done by the Gateway, there is no way to specify which storage node's status page is returned on a request.

The following addition to the policy for the domain will explicitly deny anonymous users access to "/" so that they cannot view the node status page.

```
...
{
  "Effect": "Deny",
  "Sid": "AdminPage",
  "Principal": {
    "anonymous": [
      "*"
    ]
  },
  "Action": [
    "GetObject"
  ],
  "Resource": "/"
}, ...
```

By changing the principal, the previous policy example can adapted for any user or group for which you wish to deny access to the node status page.

Gateway Operations

- Service Control and Status
- Java Runtime Parameters
- Server Firewall
- Create Domains and Buckets

Service Control and Status

To manually control the start-up and shutdown of the Gateway service and to get its running status, use the following commands:

RHEL/CentOS 7

```
systemctl start cloudgateway
systemctl stop cloudgateway
systemctl status cloudgateway
```

Java Runtime Parameters

The `/etc/sysconfig/cloudgateway` file contains the Java process memory settings and the maximum number of open file descriptors for Gateway.

Default Java Process Settings

```
HEAP_MIN=1024m
HEAP_MAX=1024m
MAX_OPEN_FDS=25000
```

Memory: While the default JVM heap memory settings will work well for a small deployment, you should increase them for larger deployments that handle a large transaction load for multiple tenants. The JVM heap memory is utilized for caching frequently used objects, authentication results, and other operational data.

Maximum file descriptors: While the default maximum number of open file descriptors will work well for a small deployment, you should increase the limit for larger deployments that handle a large transaction load for multiple tenants. Network socket connections for the upstream clients plus the back-end connection pool will comprise the majority of the open file descriptors during Gateway operations.

> Important
>
> With CentOS 6, verify that Gateway's MAX_OPEN_FDS value is allowed: run "`sudo ulimit -H -n`", which should be larger than MAX_OPEN_FDS. If not, increase the operating system's limits by updating `/etc/security/limits.conf`.

Server Firewall

The firewall rules from the default RHEL/CentOS installation need to be changed in order to allow inbound client access to Gateway. You can adjust the IPTABLES rules to allow inbound access for each front-end protocol or you can disable IPTABLES entirely. Execute the following commands if you wish to disable the operating system's firewall.:

RHEL/CentOS 6

```
chkconfig iptables off
service iptables stop
```

RHEL/CentOS 7

```
systemctl disable firewalld
systemctl stop firewalld
```

While it is valid to use IPTABLES in conjunction with Gateway, the service startup script will issue a notice if IPTABLES are enabled as a reminder since their use can be a source of confusion if inbound traffic to Gateway is blocked. If you have customized the inbound rules to allow access, you can safely ignore this startup notice.

Create Domains and Buckets

To create and manage domains and buckets from the Content Portal, see Configuring Domains and Configuring Buckets.

If you need to create them manually (from the command line), see Manually Creating and Renaming Domains.

Replicating Domains to Other Clusters

Replication of domains between Swarm clusters provides for disaster recovery and locality of access to content. Many replication strategies are supported by Swarm including single direction roll-up, multi-master, and cascading topographies. This section focuses on the content that must be replicated in order to host storage domains in multiple clusters.

> Important
>
> Regardless of the replication strategy selected, it is crucial that a domain, whether an administrative domain or a storage domain, is only created one time.

If a domain with the same name is created using an SCSP operation or with the `initgateway` command in multiple Swarm clusters, this actually creates different domains that share the same name. Due to the name collision, this will lead to incorrect results if the different domains are ever replicated into the same cluster. The simple rule is to only create a domain name one time and only use a Replication Feed to copy it into other clusters.

If you intend to use a storage domain within a cluster other than the one in which it was initially created, you must ensure that the administrative domain is also available in that other cluster. To use a storage domain means that client requests (read, write, delete) are being performed in the cluster. If the storage domain is replicated to another cluster purely for DR and there will never be any client requests sent to it, replication of the administrative domain is unnecessary. You must only use the `initgateway` command in one cluster and then use Replication Feeds to duplicate the administrative domain into all other clusters.

Required permission

When replicating through Gateway, Swarm needs to make a "GET /" request during replication in order to check the Swarm cluster name and version.

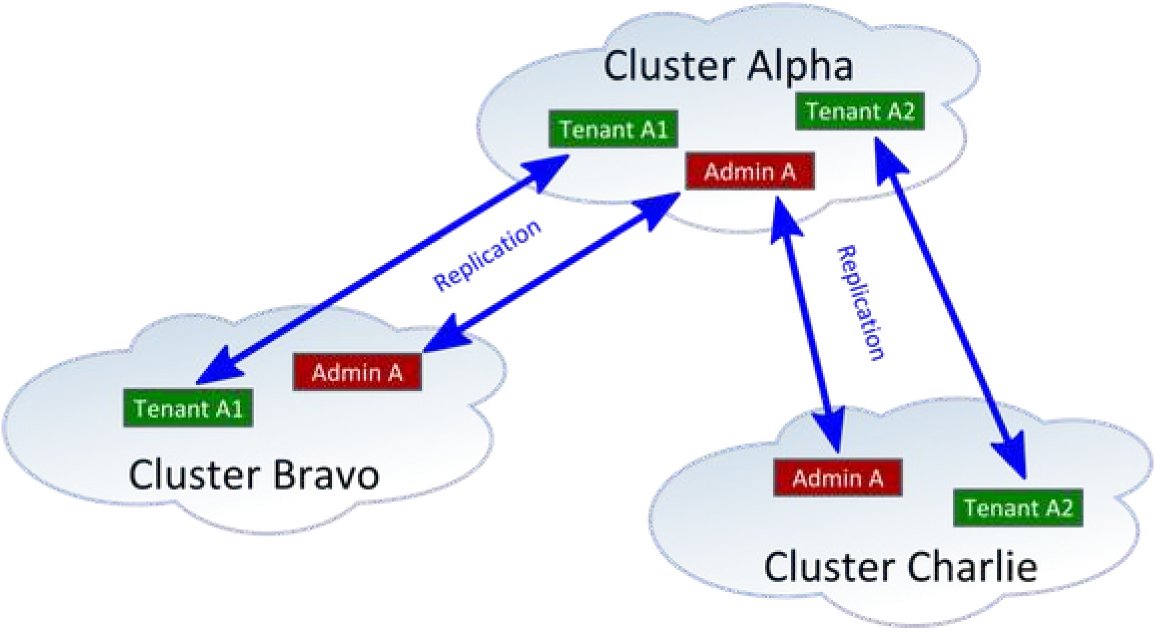
To enable Swarm to make this check, add a policy.json rule giving "anonymous" permission to "GET /" (GetObject):

```
{
   "Effect":"Allow",
   "Sid":"Swarm Node Status",
   "Principal":{
      "anonymous":[
         "*"
      ]
   },
   "Action":[
      "GetObject"
   ],
   "Resource":"/"
},
```

> See Policy Document.

Example replication

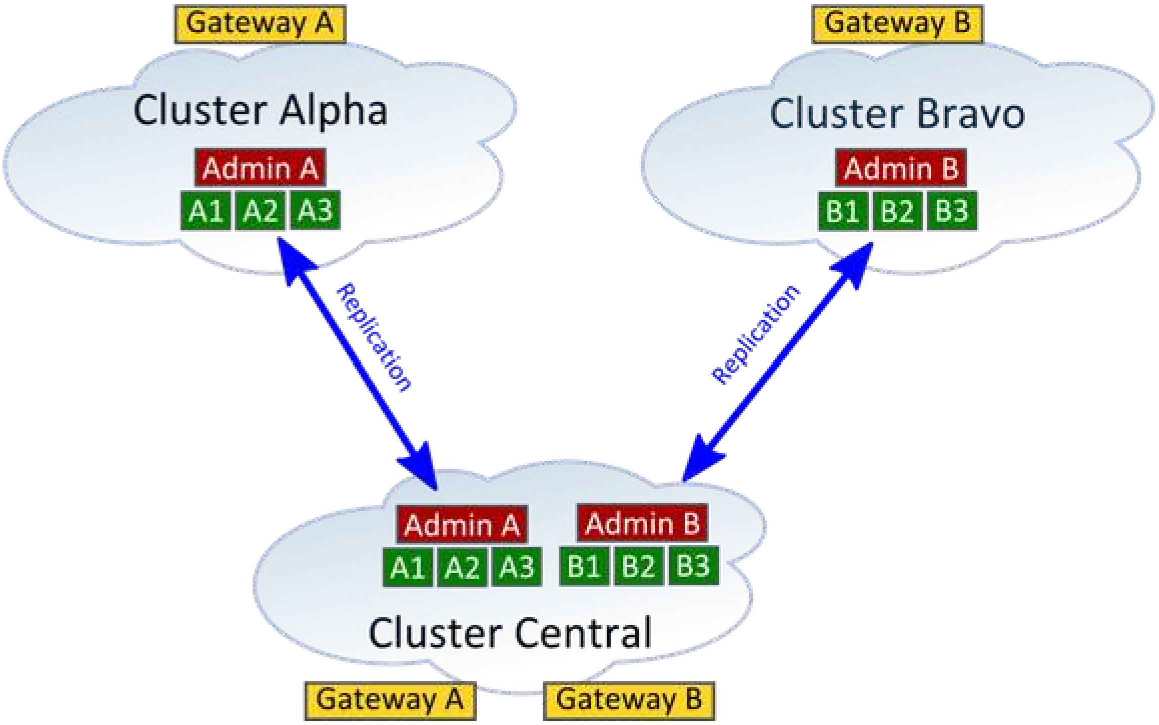
The following diagram shows three Swarm storage clusters, storage domains A1 and A2, and an administrative domain A. Domains A, A1, and A2 were all initially created in Cluster Alpha. Remote replication was then configured to mirror domains A and A1 to Cluster Bravo. Additionally, domains A and A2 were configured to mirror to Cluster Charlie.



By configuring the domains to mirror, bi-directional replication, each cluster can support a Content Gateway configured to use domain A as their administrative domain. With Gateways deployed in all three clusters and all using administrative domain A, clients may access storage domain A1 from Cluster Alpha and Cluster Bravo. Similarly, clients may access storage domain A2 from Cluster Alpha and Cluster Charlie.

By mirroring the storage domains, content will be available for reading from all of the clusters to which it is mirrored and content changes (create, update, or delete) will propagate to the other clusters. Notice that the administrative domain for a storage domain must also be mirrored to any cluster where the clients will access the storage domain. This is required so that the IDSYS, Policy, XFORM, authentication tokens, and other tracking information used to manage the storage domain are available to the Gateways running in the other clusters. Although replication of the administrative domain is not required if clients will never access a storage domain from another cluster, it is recommended in order to simplify DR if it should become necessary to restore.

It is also possible to replicate storage domains that have different administrative domains into the same Swarm storage cluster while providing client access to all storage domains.



When a Swarm storage cluster hosts multiple sets of storage domains and their corresponding administrative domains, client access is enabled by deploying a Content Gateway server for each set of storage domains. Although a Gateway can use only one administrative domain, it can access any storage domain that is associated with that administrative domain as long as the storage domain exists in the local Swarm storage cluster. The previous diagram shows storage domains A1, A2, and A3 that can be accessed through Gateway A in both Cluster Alpha and Cluster Central. The deployment of Gateway A in both clusters makes use of the mirrored administrative domain A in order to manage the mirrored storage domains.

When remote replication is being used as described, the client use cases and application architecture must account for replication latency. Following the creation, update, or deletion of content in one domain, there will be a time delay before the change is observed within another cluster. This latency depends upon the inter-cluster bandwidth and the total replication workload between the Swarm clusters.

Content Metering

- Configuring Metering
- Metering Statistics
- Metering API
- Example Metering Requests

- Index Generation for Metering
- Retaining Data for De-provisioned Resources

Gateway's metering is a scalable, flexible, integrated usage metering solution that makes use of Elasticsearch for data storage, management, and analysis. You configure usage metering to send batched storage and network statistics to your Elasticsearch server at whatever interval you need. You then access the metering data by querying Elasticsearch directly or through the Content Management API. Metering gathers the data that you need to manage the business of your organization:

- Current usage numbers let you evaluate usage quotas.
- Usage over time lets you generate billing.
- Historical usage queries populate graphs for easy monitoring from the dashboard.

> Note
> Metering replaces the legacy CSMeter package (`csmeter` and `cshistory` utilities), which is deprecated and no longer included with the Gateway.

Configuring Metering

Metering requires minimal configuration to implement:

- metering.enabled (disabled by default)
- storage_cluster.indexerHosts (must be defined for metering)

Gateway includes other configuration parameters that are specific to controlling metering for special cases.

> See Gateway Configuration.

Metering Statistics

Gateway emits two types of usage statistics: storage and network.

Type	Statistic	Description	Notes
Storage	bytesSize	Sums the bytes of content (logical objects, including versions) that have been uploaded for storage in the context. Summing the individual `Content-Length` headers of the objects gives this value.	Swarm storage usage statistics are reported by bucket, domain, and tenant. Untenanted domains are grouped into a synthesized "_system" tenant. The context is a domain or a domain and bucket. The absence of a bucket returns the storage used by unnamed objects.
	bytesStored	Sums the bytes of space used on disk by all of the Swarm objects in the context, including all replicas and erasure-coded segments. This is also commonly called raw storage. This statistic yields a value that is the expected number of replicas in the cluster and does not account for temporary under- or over-replication that may exist in the cluster.	

	objectsStored	Counts the number of unique objects being stored in the context.	
Network	bytesIn	Sums the bandwidth usages from clients to the Gateway.	Network usage is reported at the context to which the requests are made, which means the particular bucket or domain . Network usage only includes storage operations and excludes Management API requests.
	bytesOut	Sums the bandwidth usages from the Gateway to clients.	
	opCount	Counts the number of client operations.	

Metering API

The API for metering is part of the Gateway's Content Management API.

Request

Each tenant, domain, and bucket has a subresource prefix, meter:

- Tenant t1: /_admin/manage/tenants/t1/meter/
- Domain d1: /_admin/manage/tenants/t1/domains/d1/meter/
- Bucket b1: /_admin/manage/tenants/t1/domains/d1/buckets/b1/meter/

Under meter, this is the endpoint for the specific context (tenant, domain, bucket):

Metering endpoint for context

```
meter/usage/{metric}
 ?from={startDate}
 &to={endDate}
 &groupBy={groupBy}
```

Value	Required	Case	
{metric}	Yes	Case-sensitive	Specifies which metric to analyze: • bytesIn (from client to Swarm) • bytesOut (from Swarm to client) • bytesSize (sum of logical objects' Content-Length values) • bytesStored (sum of physical disk storage consumed) • objectsStored (number of logical objects) • opCount (operation count, minus Management API requests)

{startDate}	Yes		`YYYY-MM-DDT00:00Z`
			`YYYY-MM-DDThh:mmZ`
			`YYYY-MM-DDThh:mm:ssZ`

{endDate}	Yes		`YYYY-MM-DDT00:00Z`
			`YYYY-MM-DDThh:mmZ`
			`YYYY-MM-DDThh:mm:ssZ`
			Must be later than {startDate}.

{groupBy}	No	Case-sensitive	Specifies which time increment to group by: - hour - day - unspecified - no grouping: date range is collapsed to a single value Note Group by aggregates metrics using `average` for storage metrics and `sum` for network metrics.

To fetch a metric for the children of a context (either the tenant's domains or the domain's buckets), add **`/children`** :

Metering endpoint for child of context

```
meter/usage/{metric}/children
 ?from={startDate}
 &to={endDate}
 &groupBy={groupBy}
```

Specific to storage metrics only, you can run a point-in-time query:

Point-in-time storage metrics

```
meter/usage/bytesStored/current
meter/usage/objectsStored/current
meter/usage/bytesStored/current/children
meter/usage/objectsStored/current/children
```

Note
For all current metrics, no date range is required and grouping is not applicable.

Response

The response to a query is an array of objects ("rows"), with fields that correspond to the data for each entry. These are the possible fields:

tenant domain bucket	The name of the applicable tenant, domain, or bucket for the object. Untenanted domains are grouped within the "_system" tenant name. Unnamed objects in a domain are grouped within an empty string ("") bucket name. The domain can also be an empty string, recording requests at the tenant level outside of any domain. If the domain or bucket had activity during the requested timeframe, but the name is not available because it has been deleted, the UUID is returned instead. The UUID corresponds to the former domain or bucket's `Castor-System-Alias` value.
bytesIn bytesOut bytesSize bytesStored objectsStored opCount	The value for the metric requested, which corresponds to the {metric} from the request.
timestamp	For queries grouped by time, the timestamp for a given time grouping. For example, if you are grouping by day across a week of time, the timestamp identifies which of those 7 days relates to each result.

If no records exist for the query range, an empty list ("`[]`") will be returned. When fetching the children of a context, if a child has no data for the query range, its record will be excluded from the response.

Note
Using **`/children`** and a time grouping together might result in additional rows to express each time/child combination, just as RDBMS queries with multiple GROUP BY arguments return separate rows per every combination.

Example Metering Requests

Example request/response

Following is a result from a query for **`/children`** that uses a **`day`** grouping. The target of this query is the domain `domain1`, which belongs to tenant `tenant1`.

Note that the results are for the children of the domain, which are its buckets (as opposed to the children of a tenant, which are its domains):

Example metering request/response

```
GET
/_admin/manage/tenants/tenant1/domains/domain1/meter/usage/bytesIn/child
ren
 ?from=2015-07-01T00:00Z&to=2015-07-03T00:00Z&groupBy=day
[
   {
       tenant: "tenant1",
       domain: "domain1",
       timestamp: "2015-07-01T00:00:00.000Z",
       bucket: "research",
       bytesIn: 27277
   }, {
       tenant: "tenant1",
       domain: "domain1",
       timestamp: "2015-07-01T00:00:00.000Z",
       bucket: "archive",
       bytesIn: 18771
   }, {
       tenant: "tenant1",
       domain: "domain1",
       timestamp: "2015-07-02T00:00:00.000Z",
       bucket: "research",
       bytesIn: 27855
   }, {
       tenant: "tenant1",
       domain: "domain1",
       timestamp: "2015-07-02T00:00:00.000Z",
       bucket: "archive",
       bytesIn: 19645
   }
]
```

Common billing queries

These are queries that are common when integrating with billing systems where charges for bandwidth in/out and storage are calculated at the end of a calendar month. In these examples, the period being queried is Midnight 2016-06-01 UTC through Midnight 2016-07-01 UTC. Note that the storage numbers are the average storage over the month while the bandwidth is the total at the end of the month.

System tenant raw storage by domain

```
GET /_admin/manage/tenants/_system/meter/usage/bytesStored/children
   ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Tenant 'bravo' raw storage

```
GET /_admin/manage/tenants/bravo/meter/usage/bytesStored
  ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Domain 'xray.example.com' raw storage

```
GET
/_admin/manage/tenants/delta/domains/xray.example.com/meter/usage/bytesS
tored
  ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Bandwidth IN for domains of tenant 'tango'

```
GET /_admin/manage/tenants/tango/meter/usage/bytesIn/children
  ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Bandwidth OUT for domains of tenant 'tango'

```
GET /_admin/manage/tenants/tango/meter/usage/bytesOut/children
  ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Raw storage for domain 'uniform.example.com' by bucket

```
GET
/_admin/manage/tenants/tango/domains/uniform.example.com/meter/usage/byt
esStored/children
  ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Logical storage for domain 'uniform.example.com' by bucket

```
GET
/_admin/manage/tenants/tango/domains/uniform.example.com/meter/usage/byt
esSize/children
  ?from=2016-06-01T00:00:00Z&to=2016-07-01T00:00:00Z
```

Index Generation for Metering

Metering uses a different index for each day, which makes it efficient to expire old data. The daily index is created by utilizing Elasticsearch’s index alias to combine the records into a queryable whole. To support this, each gateway runs a daily maintenance task that deletes any indices older than the retention period and adds the new daily index to the alias. This maintenance is scheduled with offsets to avoid having multiple gateways performing the task simultaneously.

> Note
> Statistics are batched for a period of time. The date that samples are assigned is the date when they are written, not when they were collected. A new index for the new day is created automatically.

Retaining Data for De-provisioned Resources

When you need to de-provision a tenant or a domain, take care not to inadvertently lose access to your metering data. When a tenant/domain is removed, there is no longer a Content Management API path to which to refer for metering queries.

Best practice - Retain the historical metering data for each decommissioned entity:

1. Change ownership of the tenant or domain to be an admin user.
2. Update the Policy to remove non-admin access.
3. Purge all of the objects being stored in the decommissioned domain (or all the domains for a decommissioned tenant).
4. Important: Do not delete the empty tenant or domain; retain it as is.

Once you no longer need to retain the historical usage data, you can delete the decommissioned tenant or domain. Once deleted, the domain or bucket name is not available but it might still be returned in metering queries covering an earlier time. Its UUID will be returned instead of its name.

Gateway Troubleshooting

This section provides a starting point for troubleshooting operational issues with your Gateway deployment and integrated applications.

- Bad IDSYS or Policy
- Domain and Bucket Ownership
- LDAP Configuration
- Portal Upload Error
- Unexpected HTTP Responses
- Unexpected HTTPS to HTTP Redirects

Bad IDSYS or Policy

If you write an incorrect IDSYS to a tenant or a storage domain, subsequent attempts to access the system will return a 503 error. This is an example response:

```
HTTP/1.1 503 Service Unavailable
 Server: CAStor Cluster/6.5.4
 Via: 1.1 172.16.99.70 (Cloud Gateway SCSP/2.2)
 Gateway-Request-Id: D9DF0347CB7EAAE9
 Gateway-Error-Message: Unable to connect to identity system
     ldap://172.16.99.20:636 as cn=gateways,dc=caringo,dc=com:
     javax.naming.ServiceUnavailableException: 172.30.0.42:636; socket
closed
 Content-Length: 44
 Identity system failure or misconfiguration
```

To work around this, authenticate as qualified user that is defined in the tenant IDSYS or in the root IDSYS and replace the bad IDSYS. For example, if the user admin exists in the root IDSYS, write a corrected version of the storage domain's IDSYS and authenticate the request as user "`admin@`".

> Warning
> If you write an incorrect Policy to a tenant or storage domain, you could lock yourself out.

Workaround

To work around a Policy problem, authenticate with a "!" prefix on the user name and replace the bad Policy. For example, if the user admin exists in the root IDSYS and the storage domain's Policy has denied access to all users, write a corrected version of the Policy to the storage domain by authenticating as "`!admin@`".

> For more information, see the IDSYS Document Format.

Domain and Bucket Ownership

- Get a domain's metadata

- Change the owner of the domain
- Get a bucket object's metadata
- Change the owner of the bucket
- Change to anonymous ownership

In order to troubleshoot access permission problems related to ownership for domain or bucket contexts, it can be helpful to query Swarm directly. Check that the context (domain or bucket) owner has the correct value.

> Unknown Tenant
>
> If Gateway gets a request to a domain that is marked as belonging to an unknown tenant, it handles the request as if it was from the "System Tenant" and provides troubleshooting guidance in the application logs.

> Note
>
> For these examples, the string `{node-IP}` represents the IP address of any Swarm node in the storage cluster. These examples assume that you have direct access to the storage nodes without going through the Gateway.

Get a domain's metadata

```
curl -I -L http://{node-IP}/?domain=dom1.example.com'
```

Change the owner of the domain

```
curl -X PUT -L
 --post301
 -d ''
 -H 'X-Owner-Meta: ccarlin' 'http://{node-IP}/?domain=dom1.example.com'
```

Get a bucket object's metadata

```
curl -I -L 'http://{node-IP}/bucket1?domain=dom1.example.com'
```

Change the owner of the bucket

```
curl -X PUT -L
 --post301
 -d ''
 -H 'X-Owner-Meta: ccarlin'
'http://{node-IP}/bucket1?domain=dom1.example.com'
```

Note that in these examples, you would also need to include any custom metadata for the domains or buckets in the updates. With the curl command, this done with multiple -H arguments.

Change to anonymous ownership

To change a domain or bucket to anonymous ownership so that everyone has full access to it, remove the X-Owner-Meta header or assign it a blank value with: -H 'X-Owner-Meta: '

LDAP Configuration

Problems with the configuration of the LDAP identity management settings can prevent user authentication and the determination of group membership. All LDAP configuration items are kept within the root IDSYS document, stored in the Gateway server's file system, and the IDSYS documents for tenants and storage domain. Start troubleshooting by:

1. Determining which IDSYS document is being used based on the format of the user name (see Content Application Development for details on the login format).
2. Ensuring that the fields in the IDSYS that is being used are correct.

After determining the IDSYS that is being used, debug basic connectivity and queries directly with the LDAP server. Look in the Gateway server's log in order to get the exact LDAP search filter that it is trying to use.

This is an example from the log where it checks if a user belongs to an allowed group:

```
2012-09-13 22:25:47,671 DEBUG [qtp1355087478-37 -
/1347593126.86weirdbucket/x/foo.txt?domain=1347593126.86example.com|2155
222263AE4638]
 Policy: Searching for user in ou=groups,dc=example,dc=com
 with filter (&(objectclass=*) (memberUid=john)(|(cn=Finance)))
```

The log entry of the LDAP search filter can be converted to an LDAP URL that can be used by a tool like curl in order to query the LDAP server. The format of the URL is:

```
ldap://HOST:PORT/ROOT??sub?FILTER
```

Using the LDAP search filter information from the example Gateway log, this shows how to use the curl command to query the LDAP server directly.

```
curl
"ldap://localhost/ou=groups,dc=example,dc=com??sub?(&(objectclass=*)
(memberUid=john)(|(cn=Finance)))"
```

If the connection is successful and the query finds users with the group, the output will be similar to this:

```
DN: cn=Finance,ou=groups,dc=example,dc=com
gidNumber: 10002
memberUid: fred
memberUid: john
description: Group account
objectClass: posixGroup
cn: Finance
```

If there are errors, resolve them and update the IDSYS document with the corrections.

Portal Upload Error

Symptom — A Content Portal upload fails, producing an error similar to this:

```
org.apache.commons.fileupload.FileUploadBase$SizeLimitExceededException.
The request was
  rejected because its size (258196924634) exceeds the configured
maximum (26355234816)
```

Cause — The Gateway's spool directory has reached its maximum capacity. The spool directory is the temporary space to support HTTP multipart MIME uploads, which are the requests that are made by uploading files into the Content UI Overview.

Solution — Move the spool directory to one that has enough free space to support your largest simultaneous uploads.

In the Gateway configuration file, locate the [gateway] section and edit the value for multipartSpoolDir, which defaults to `var/spool/cloudgateway/`.

Unexpected HTTP Responses

If you get unexpected HTTP response codes while using or integrating with Gateway, use these tips to troubleshoot the cause of the responses. An example of an unexpected response would be if permission is denied (HTTP 403) on an object that you believe should be accessible.

> Tip
> Enabling Swarm Storage audit logs allows you to track all of the requests that Gateway makes that are related to a client request.

- Request ID in responses
- Request ID in logs

Request ID in responses

To aid with tracking transactions, all of Gateway's HTTP responses contain a header with the request ID. This request ID is also recorded in the Gateway's server log file and the audit log file. Searching the server log file for a given request ID will show the processing steps that took place during the handling of the request.

These are three examples where the request ID is found in a client response:

SCSP response header

```
Gateway-Request-Id: 375400C95338546F
```

S3 response header

```
x-amz-request-id: 375400C95338546F
```

S3 error response body

```
<RequestId>375400C95338546F</RequestId>
```

Request ID in logs

Search for the request ID in the Gateway server log:

```
grep "375400C95338546F" /var/log/cloudgateway/server.log
```

The search results from the log will show:

- Request URI and whether an Authorization or cookie header is on the request
- Action being performed, such as CreateDomain, GetObject
- Owner of the context for the request
- Merged Policy document used to evaluate authorization
- LDAP search filter used for user or group lookups
- Reason for the HTTP response

The merged Policy is normally a combination of the root, domain, and bucket policies. An example log entry showing the context owner and a merged Policy document is:

```
2014-03-31 11:12:32,442 DEBUG [qtp1994043452-35|C66CF2A1D4DD4C8D]
 Auth: AUTHENTICATING: 'ldap john@'
 Action is GetObject, user idsys is ldap (root), context owner is john@
and merged policy is:
 [Sid=1 Allow [AllActions] "/ *" {group=[CloudAdmins]} {}]
```

When using the Policy conditions, such as the Referer header restrictions, the merged Policy that is logged is the one that was used to evaluate permissions for the request. If you are not seeing the portion of the Policy that you expected, check the condition statements to see which is being used.

Additional error details are contained with the HTTP response header `Gateway-ErrorDetails` and are also logged in the Gateway server log. An example of this type of log message is:

```
2012-10-19 08:41:28,327 DEBUG [qtp596850781-35 -
/reports?domain=example.com| F09B3F5FCA0A477F]
 Auth: Request failed: 403 User is not allowed and is not owner.
 owner: john, user: george, dn: uid=george,ou=people,dc=example,dc=com
```

In the previous example, the user george, the full LDAP DN is given, is not allowed to perform the requested action within the /reports bucket because he is not the owner, john, and because there is no Policy that grants him permission.

If necessary for application debugging, the Gateway can dump the HTTP request headers received with each request. To enable request header logging, add the following setting to the `gateway.cfg` file and restart the Gateway process:

```
[debug]
showRequest = true
```

> Tip
>
> Use debug sparingly! It can produce a significant amount of extra information in the server log, including security-related Authorization and Cookie headers.

An example of the request headers in the log output is:

```
2013-08-30 15:23:24,804 DEBUG [qtp1872474714-40|F69277697D792B98]
 Auth: REQUEST: POST /?domain=john.example.com AUTHORIZATION
 ContentType: application/castorcontext
 workaround-content-type: application/ castorcontext
 Host: 172.10.8.5:8084
 Content-Length: 0
 Accept: */ *
 User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4
OpenSSL/0.9.8x zlib/1.2.5
 Authorization: Basic am9objpwYXNzd29yZA==
```

Unexpected HTTPS to HTTP Redirects

If you are attempting access the Content Portal through HTTPS and your browser is being redirected to an HTTP link, you may need to alter your front-end load balancer settings. This issue can occur when a load balancer blindly passes back a `Location` header from the Gateway without rewriting it to match the front-end protocol of the service.

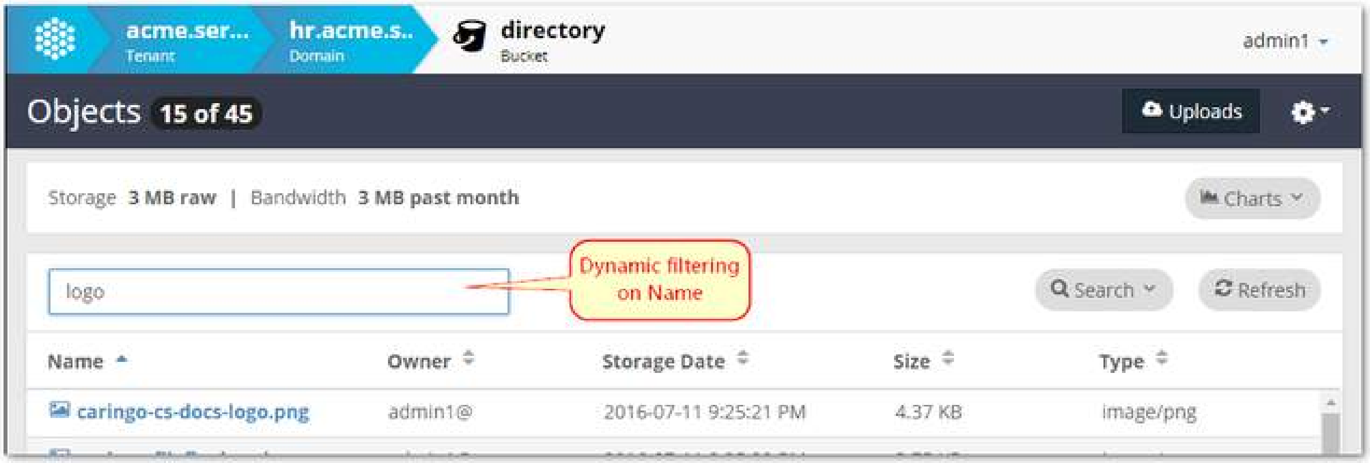
To correct for this, configure your load balancer to inject the header "`X-Forwarded-Proto: https`" with all SSL/TLS requests sent to the back-end pool of Gateway servers. This header instructs the Gateway to use HTTPS in any `Location` header that it sends to the browser.

Swarm Content UI

- Content UI Overview
- Using the Content UI
- Configuring Tenants
- Configuring Domains
- Configuring Buckets
- Setting Quotas
- Setting Storage Policies
- Setting Permissions
- Setting Tokens
- Uploading Files
- Managing Collections
- Downloading Content
- Metadata Encoding
- Editing Names, Metadata, and Versions
- Usage Reports

Content UI Overview

The Content UI offers a visual representation of the cluster, organized by tenants, domains, collections, and buckets. Depending on the level of the login credentials (Root, Tenant, or Domain) and the access policy in force, the Content UI displays only the information authorized; for example, domain-level users can at most view, add, or update buckets and bucket contents.



Name	Owner	Storage Date	Size	Type
caringo-cs-docs-logo.png	admin1@	2016-07-11 9:25:21 PM	4.37 KB	image/png

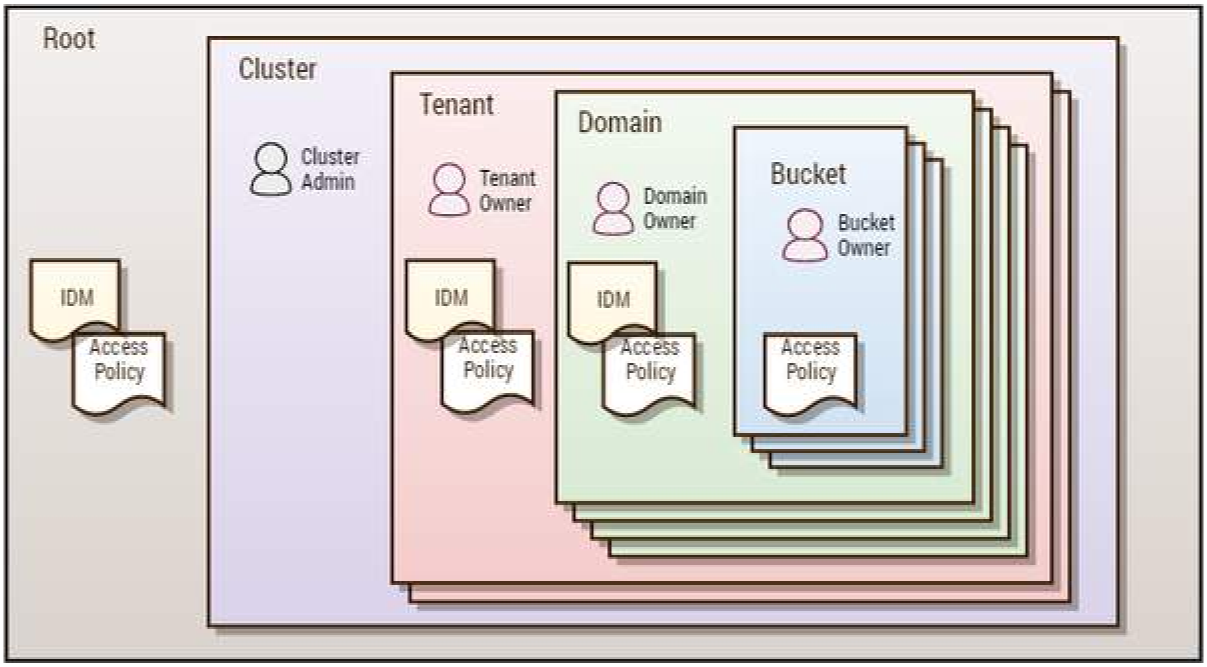
The Content UI layers on top of Gateway to provide a user interface for these key activities:

- Tenant creation and management, including definition of authentication configuration and access policies
- Domain creation and management, including definition of authentication configuration and access policies
- Content creation and listing capabilities, including bucket creation, browse or drag-and-drop content uploading, bucket listing, link sharing, and content downloading
- Content search of domains and buckets, by name, size, owner, date, type and custom metadata

> See Content Gateway Concepts.

Understanding Scope

This diagram shows the hierarchy of the scopes (cluster, tenant, domain, bucket) within Gateway and a Swarm Storage cluster:



A cluster can contain multiple tenants. Tenants can each contain multiple storage domains. Storage domains are known to Swarm and are where content is stored.

Caringo only defines the role of owner. You can create simple to sophisticated role-based access control (RBAC) definitions as required for your organization using Gateway's access control policies. Note that the Cluster, Tenant, Domain, and Bucket "admins" shown in the diagram above are typical roles, but they are not required or hard-coded into the system.

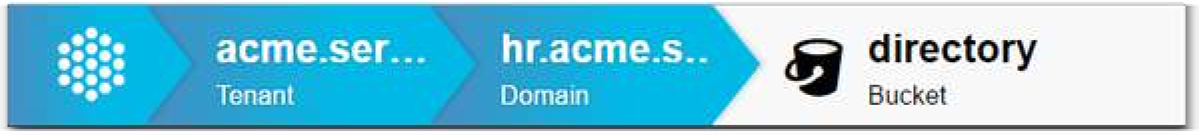
Using the Content UI

- Accessing the Content UI
- Creating a tenant
- Creating a storage domain
- Creating a bucket
- Uploading content
- Creating a search collection
- Resources

The Content UI lets you create, view, and manage your tenants, domains, buckets, and objects from the convenience of a web browser.

Tip
For the best experience using the Content UI, use the latest Chrome or Firefox browsers.

A cluster can contain multiple tenants. Tenants can each contain multiple storage domains. Storage domains are where content is stored.



Important
What is visible in the Content UI is controlled and protected by access permissions. A Domain admin, for example, will not be able to see anything (domains, tenants, clusters) outside of the domain that they are authorized to manage.

The cloud icon in the breadcrumb trail attempts to take users to the top-most level (domain, tenant, or root) allowed for them by their permissions. It takes them to the storage domain they logged into (the URL in the address bar); however, if the hostname is not a storage domain, then the user must go to the root level, even if their permissions do not allow them to list.

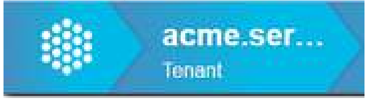
> See Content Gateway Concepts.

Accessing the Content UI

The admin users for the Content UI are defined during the Gateway installation, via a root policy configuration file that grants full cluster permissions to a specific LDAP group or list of users.

> Note
> You must login as an authorized user. There is no anonymous access to the Content UI.

Cluster	To log in as a cluster administrator, browse to the cluster's Content UI or a storage domain and use the login name and password created for the Root administrator:
	`http://{gateway·IP}/_admin/portal/`
	`http://{storage·domain}/_admin/portal/`
	The Content UI opens to the list of tenants or, if that storage domain exists, to the domain's page.

Tenant or Domain	To log in as a tenant or domain administrator, browse to a domain's Content UI and log in with your admin credentials: `http://{storage·domain}/_admin/portal/` The Content UI opens to the domain's page, from which you can access the tenant's information (if allowed) by clicking the breadcrumb menu.  acme.ser... Tenant
Different Tenant or Domain	To log into a different tenant or domain with your existing credentials, specify your tenant or domain after your user name: ● Tenant – append a plus '+' sign and the tenant name: `username+yourtenant` ● Domain – append an at '@' sign and the domain name: `username@yourdomain.com`

Creating a tenant

1. From the list of Tenants, click the Add button.
2. Type in the name of the new tenant. (See Naming Rules for Swarm.)
3. Press Enter or click the Add button again to save it.

> Note
> The SYSTEM TENANT has no owner or configuration options; it is the permanent system-created tenant that manages any storage domain that is not associated with another tenant.

> See Configuring Tenants.

Creating a storage domain

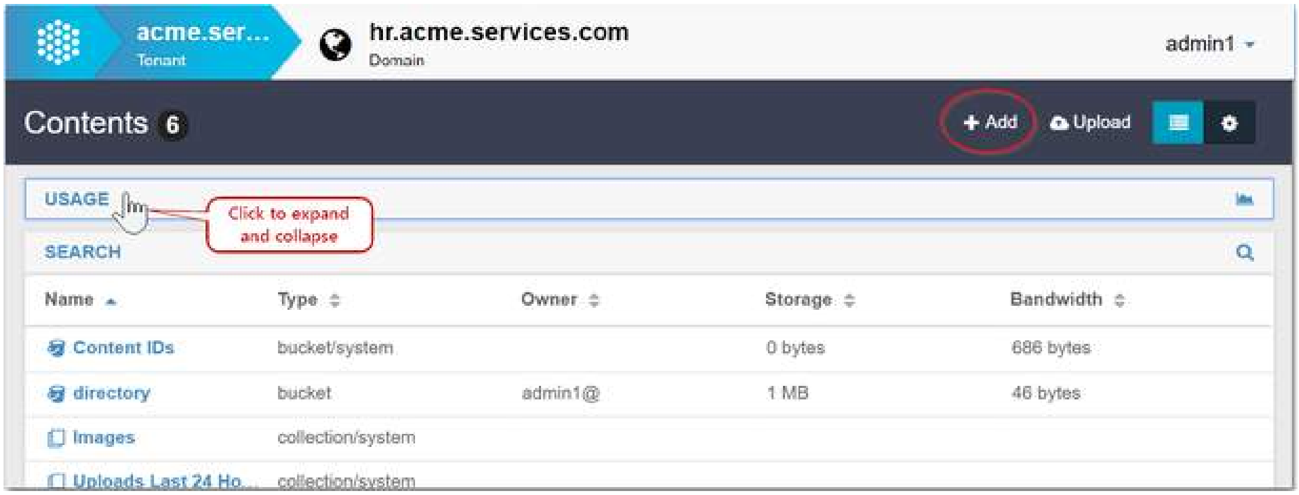
1. Click on a tenant to view the list of domains within it.
2. To create a domain, click the Add button.
3. Type in the name of the new domain. (See Naming Rules for Swarm.)
4. Select Add to save it.

> See Configuring Domains.

Creating a bucket

1. Click on a domain to view the contents within it.

> Tip
> Click any blue-text title bar, such as USAGE, to collapse it down to view more content listings. Click again to expand it.



Click to expand and collapse

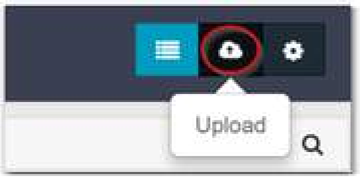
Name	Type	Owner	Storage	Bandwidth
Content IDs	bucket/system		0 bytes	686 bytes
directory	bucket	admin1@	1 MB	46 bytes
Images	collection/system			
Uploads Last 24 Ho...	collection/system			

2. To create a bucket, click the Add button.
3. When prompted for the container type, select Bucket.
4. If you need to use a non-S3-compliant name, clear the S3 Compatible checkbox.
5. Type in the name of the new bucket. (See Naming Rules for Swarm.)
6. Select Add to save it.

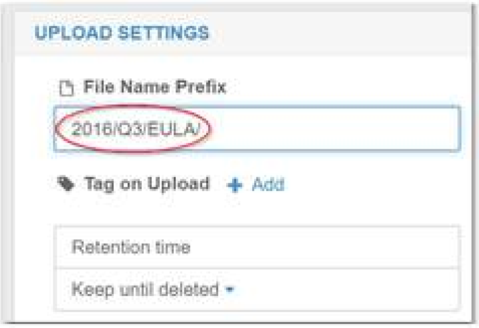
> See Configuring Buckets.

Uploading content

1. From the bucket, click the Upload button.



2. (optional) Click the blue-text title bar UPLOAD SETTINGS to set custom options for file naming, tagging, and retention policy:



UPLOAD SETTINGS

File Name Prefix

2016/Q3/EULA/

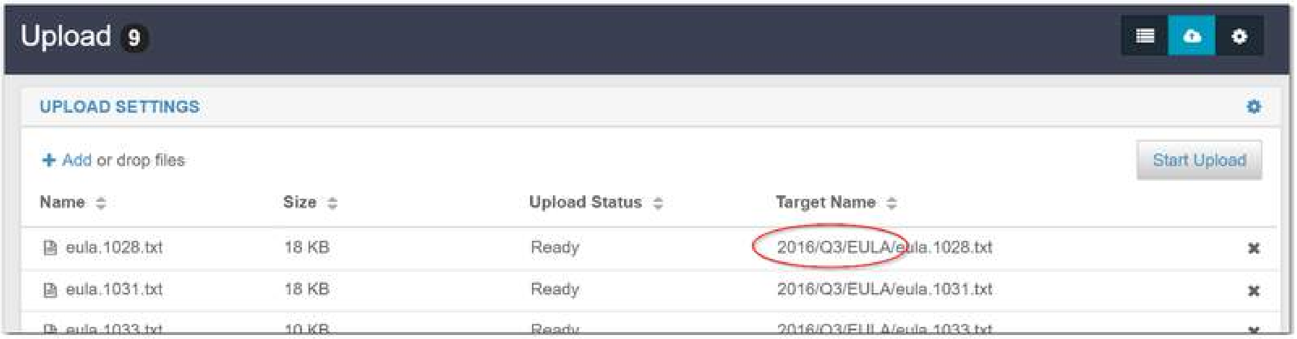
Tag on Upload + Add

Retention time

Keep until deleted

- File Name Prefix - can include one or more slashes (named objects only).
- Tag on Upload - lets you add custom metadata headers.
- Retention time - for use if you do not want the default (Keep until deleted).

3. Click Add to browse to local files or drag and drop them directly onto the upload area to queue them for upload:



Upload 9

UPLOAD SETTINGS

+ Add or drop files

Start Upload

Name	Size	Upload Status	Target Name	
eula.1028.txt	18 KB	Ready	2016/Q3/EULA/eula.1028.txt	✖
eula.1031.txt	18 KB	Ready	2016/Q3/EULA/eula.1031.txt	✖
eula.1033.txt	10 KB	Ready	2016/Q3/EULA/eula.1033.txt	✖

Tip

Check the Target Name column to validate the final object names before you start the upload. There is no name prefix option available when uploading files as Content IDs.

4. Click the Start Upload button to launch the upload with these settings.

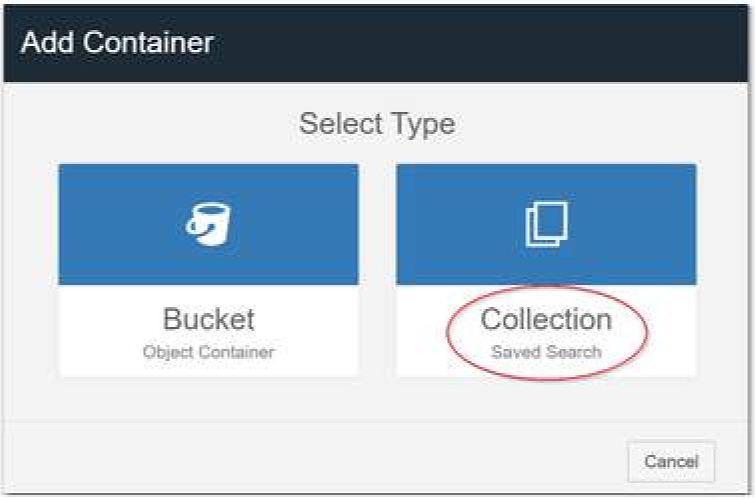
> See Uploading Files.

Creating a search collection

When you add a Collection, you are defining and running a new search. After you view the results, you can save it as a named collection, for future use.

1. Click on the domain name in the breadcrumbs bar to return to the domain.
2. To create a collection (which is a saved search), click the Add button.

3. When prompted for the container type, select Collection.

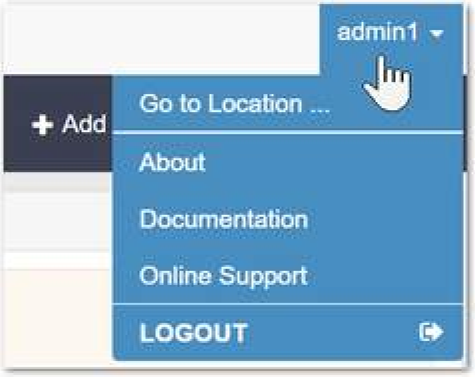


4. Search for some of the data you just uploaded, and click the Refresh button to rerun your search:
   - Filter - for string searches on object names
   - Search Scope - search the entire domain, just unnamed objects (Content IDs), or a specific bucket.
   - Add - specify a new search filter, such as Type equal to the string `image`.
   - Column Headers - customize the view the list of matched content on the bottom half of the screen.
5. To save the search as a collection for future use, click the Save button at top right.
6. Enter the name for the new collection and click Save.

> See Managing Collections.

Resources

Located at the top right of the Content UI is your account name, which drops down a menu of resources:



- Go to Location ... – opens a window for quick navigation to resources by name
- About – reports the version of the software in use
- Documentation – opens searchable online help (requires a login to Caringo Connect)
- Online Support – opens the Caringo Support site
- Logout – ends the current session

> See Go to Location.

Naming Rules for Swarm

Follow these rules for naming the objects that you create for storage in Swarm:

Type	Reference	Rules and Notes	Examples
Tenant	RFC 1034	Applies to Gateway only. A tenant must follow the naming rules of a domain.	

Domain	RFC 1034	For maximum compatibility, ensure that your domains are valid DNS names that resolve in your network.
		A domain name must
		• Be a 7-bit ASCII byte sequence.
		• Be case-insensitive.
		• Begin with an alphanumeric character.
		• Use alphanumeric characters, underscore (_), period (.), and hyphen (-).
		• Not have adjacent or final hyphens or periods (--, .., -., .-).
		• Not be an IPv4 or IPv6 IP address.
		• (S3 compatibility) Not be longer than 253 characters.
			Valid:
			`my-cluster.example.com`
			`my_cluster.example.com`
			Invalid:
			`domain`
			`cluster_example_com`

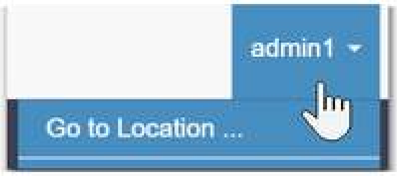
Bucket	RFC 1034	A bucket name (which is only used in the path) must • Be unique within the domain. • Be case-sensitive. • Be a valid URL-encoded, UTF-8 byte sequence. • Content UI: URL encoding is taken care of by the user interface. • Not be a UUID (32 hexadecimal characters). • Not exceed 8000 characters (larger than that is not tested or supported). • (S3 compatibility) Use lowercase ASCII and DNS-compatible names not longer than 63 characters.	

Named object	RFC 3986	An object name must • Be unique within the bucket. • Be case-sensitive. • Be a valid URL-encoded, UTF-8 byte sequence. • Content UI: URL encoding is taken care of by the user interface.	Valid: `Accounting/Customer23-03/15`

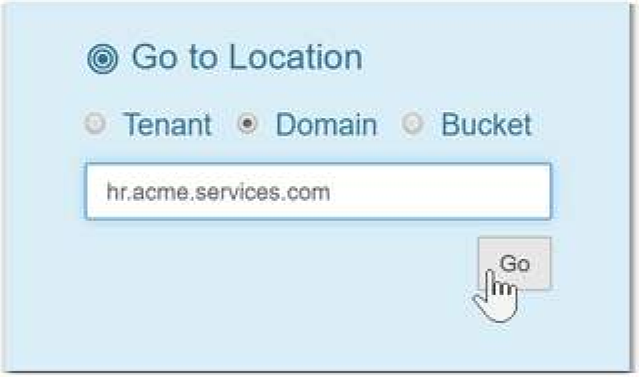
> Note
> While you may use non-ASCII characters (such as "`résumé.doc`") in bucket and object names, the URL must be properly escaped in the HTTP request ("r%C3%A9sum%C3%A9.doc").

Go to Location

The Content UI includes a quick navigation feature, Go to Location, which you access from the global resources menu that drops from your login:

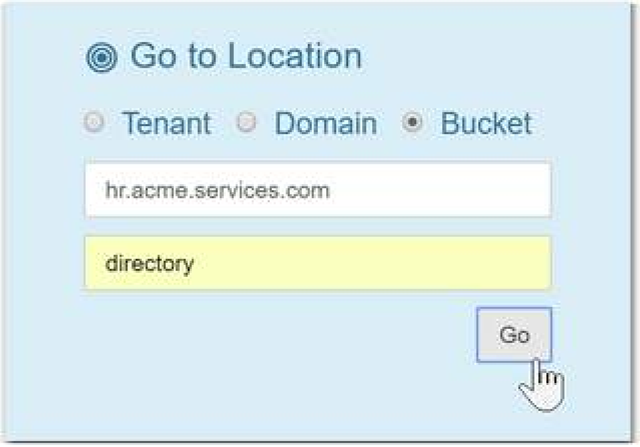


This window lets you jump directly to a specific resource (tenant, domain, or bucket) within the cluster:


Go to Location

○ Tenant ● Domain ○ Bucket

hr.acme.services.com


Go to Location

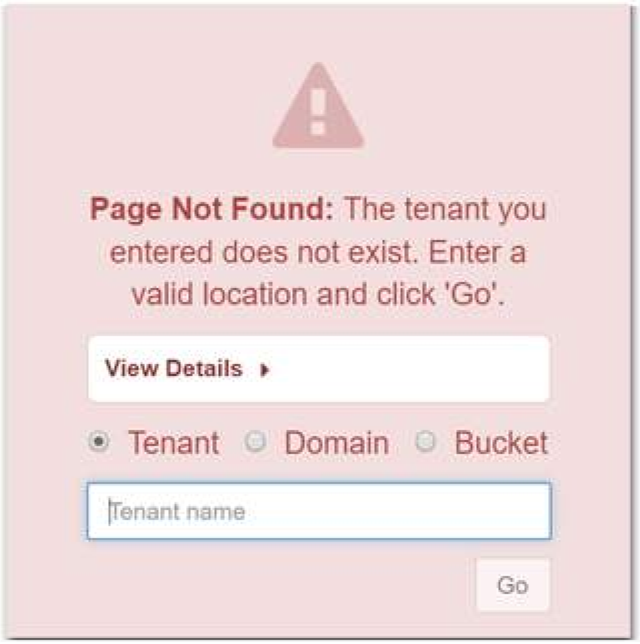
○ Tenant ○ Domain ● Bucket

hr.acme.services.com

directory

Remember that bucket names are case-sensitive. Tenants and domains are not case-sensitive; for S3 compatibility, they would be lower-case, with the exception of the special SYSTEM TENANT.

You may also see the jump from an error message, redirecting you from a location not found:



Page Not Found: The tenant you entered does not exist. Enter a valid location and click 'Go'.

Tip
If a bucket name has changed, remember to update your collections (stored searches) that reference it: select a new Search Scope, update the bucket reference, or remove the collection if it is no longer valid.

Configuring Tenants

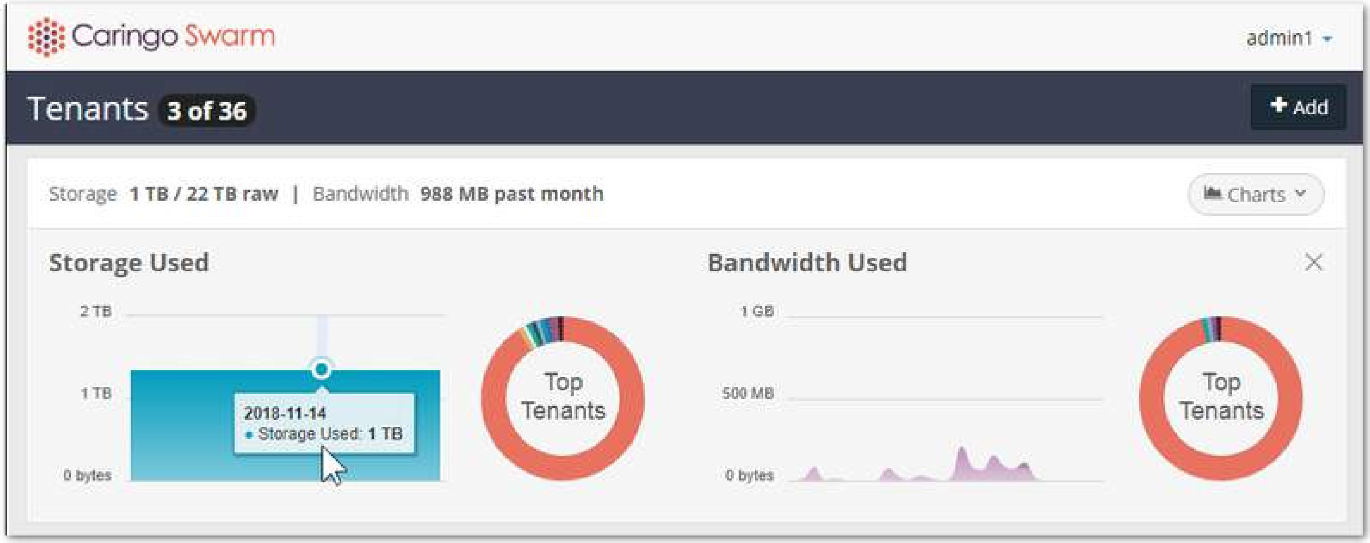
- Tenant Essentials
- Tenant Provisioning Steps
- Tenant Properties
- Permissions
- Tokens

Tenant Essentials

The concept of the tenant relates only to Gateway, not to Swarm Storage. Within a cluster, a tenant is the primary entity for dividing and controlling both access and resources. These are its critical features:

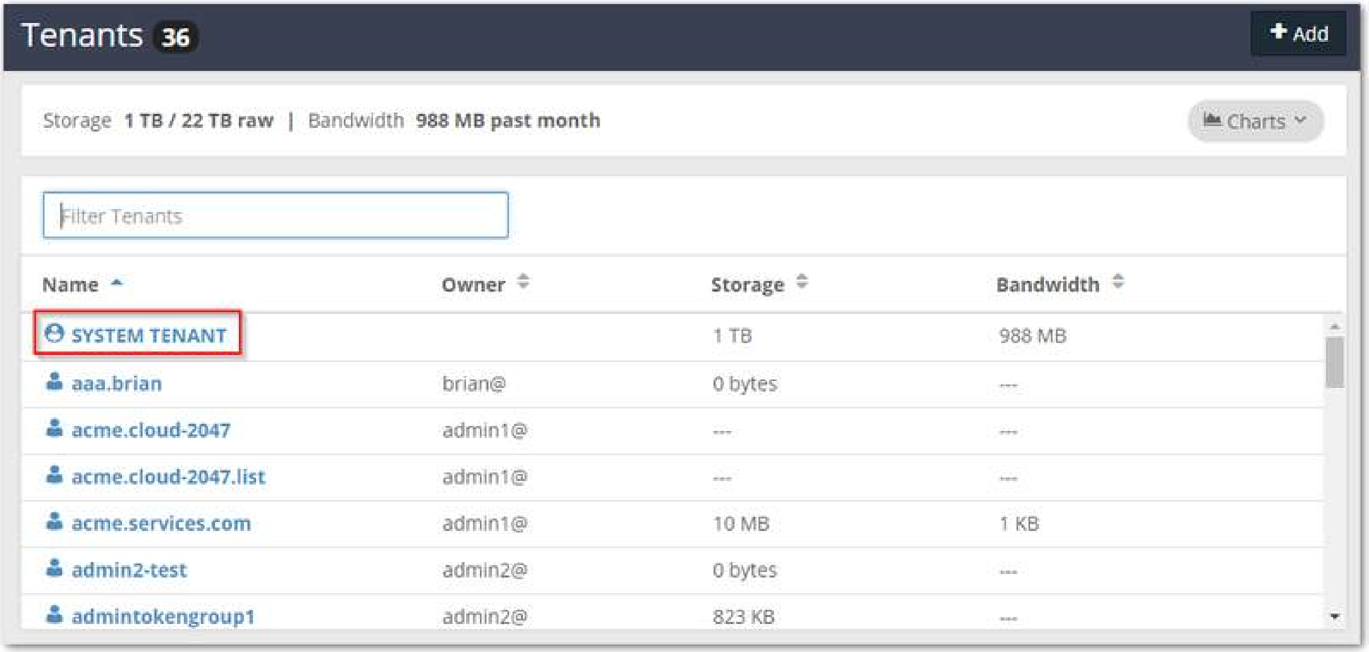
- Ownership. Each tenant owns one or more Swarm storage domains.
- Access control. Tenants can define their own identity management system so that the users and groups within them are separated from those in other tenants.
- Delegation. Tenant administrators can create and access storage domains on behalf of the tenant and they can delegate management duties for the storage domains that they create.
- No content. The tenant does not itself store end-user data; it is only a container for meta information about the tenant, its users, and its storage domains.

The Storage Used chart displays the current amount of storage used by each tenant, inclusive of all versions, replicas and erasure coded segments. The Bandwidth Used chart displays the total bandwidth (both bytes in and bytes out) used by each tenant over a rolling 30-day window.



> See Usage Reports.

All columns are sortable either ascending or descending with a default sort on the tenant name. Note the system-managed SYSTEM TENANT will always display at the start or end of the list and not in alphabetical order. If you have a large number of tenants, you can narrow the listing by entering a string in the Filter box, which filters by Name:



Name	Owner	Storage	Bandwidth
SYSTEM TENANT		1 TB	988 MB
aaa.brian	brian@	0 bytes	---
acme.cloud-2047	admin1@	---	---
acme.cloud-2047.list	admin1@	---	---
acme.services.com	admin1@	10 MB	1 KB
admin2-test	admin2@	0 bytes	---
admintokengroup1	admin2@	823 KB	---

Tenant Provisioning Steps

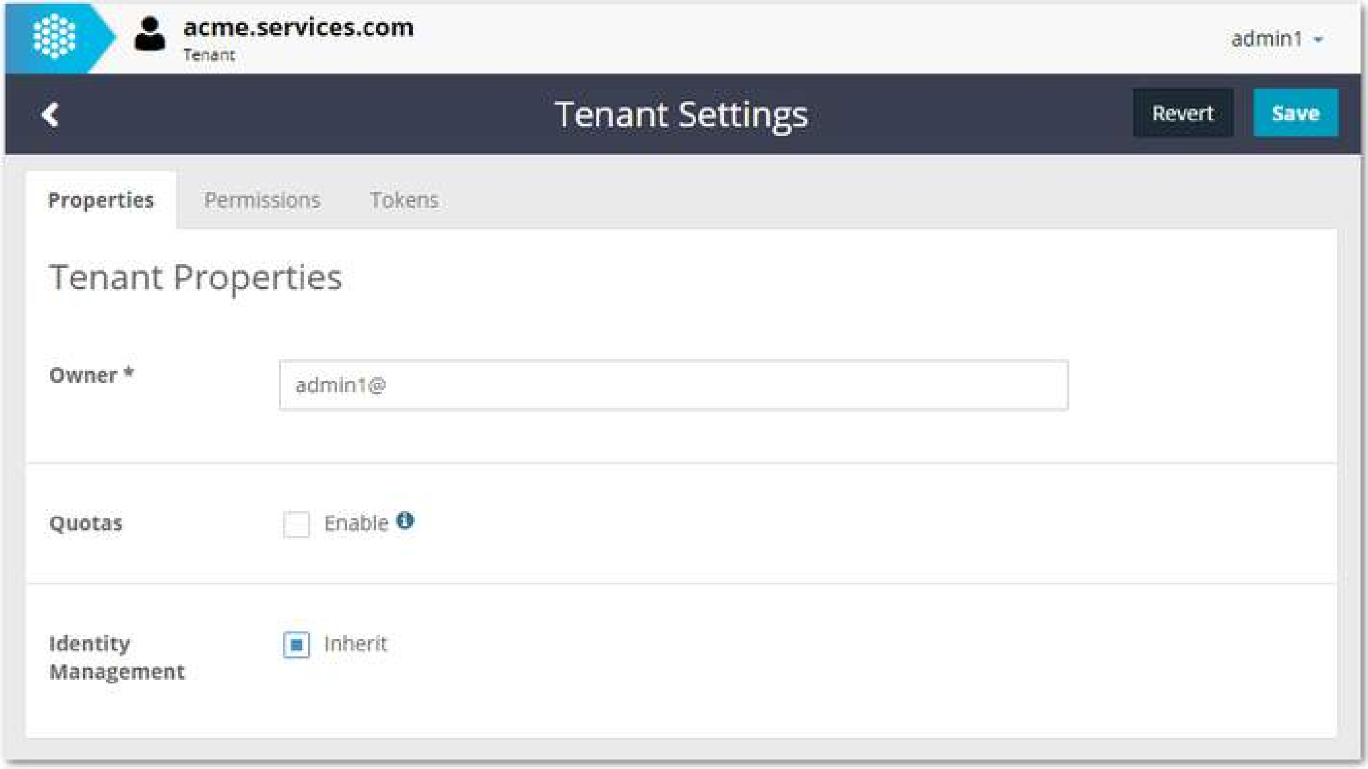
These are the typical steps when provisioning a tenant. Details for performing these steps are documented later in this guide.

1. Create the tenant.
2. Optionally,
   a. Assign ownership of the tenant.
   b. Configure the tenant's identity management system.
   c. Configure the tenant's access control policy.
   d. Configure the tenant's quota.
3. Create one storage domain within the tenant to be used by the tenant's owner or primary user.
4. Assign ownership of the storage domain to the tenant's owner or primary user.
5. Provide a login URL for the storage domain to the tenant's owner or primary user.

The format of the login URL is described in Using the Content UI.

> Important
> When delegating tenant authority to a user, remember that the user must have a starting storage domain to which to log into the Content UI.

Tenant Properties



acme.services.com
Tenant

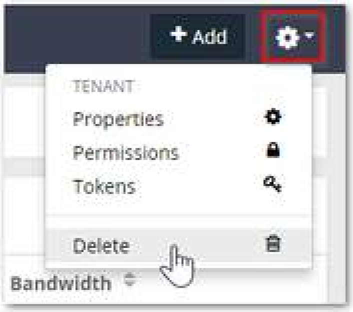
Tenant Properties

Owner * admin1@

Quotas □ Enable

Identity Management ☑ Inherit

> Important
> The Delete command deletes not only the tenant but also all of its domains, including the buckets and their uploaded contents. This command cannot be undone, so proceed with caution.



Owner

Every tenant must have an owner, who has access to and authority over its entirety. As a root admin, you can create a tenants for another to manage, just as a tenant admin can create a domain for another to manage.

> Note
> Ownership defaults to the specific root administrator who created the tenant, but the owner does not have to be a root administrator.

In general, change the owner when you are creating a context for someone else to manage. Typically, if you are creating a tenant for a customer, you do not want to own or be responsible for managing the data in that tenant.

Quotas

Quotas can be set to determine how much storage and/or network bandwidth the tenant is permitted to consume.

> See Setting Quotas.

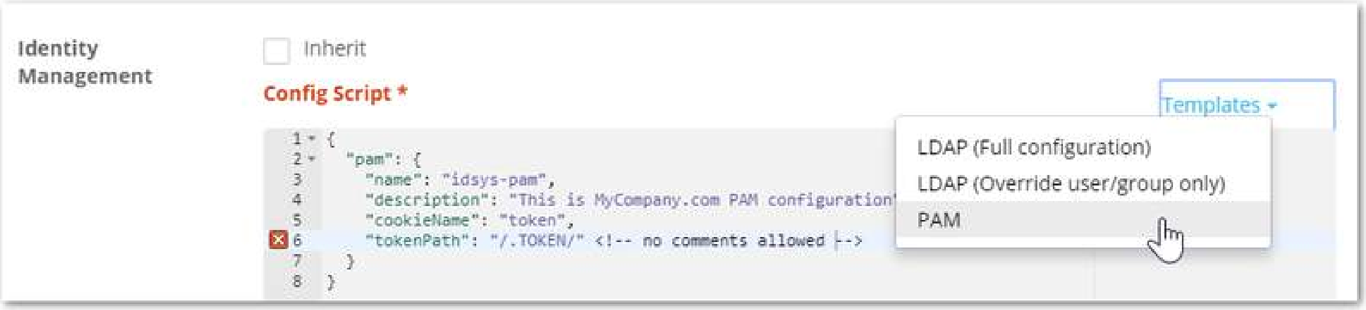
Identity Management Configuration

The IDSYS objects define the identity management systems that control the tenant's users, specifically:

- User and group information
- The authentication system

Customizing the configuration script

Every tenant inherits the root configuration, but you can disable (untick) Inherit and change your configuration.
From the Templates drop-down list, you can copy existing definitions to alter (your changes do not affect the originals).
Your script is validated in real time:



Identity Management

☐ Inherit

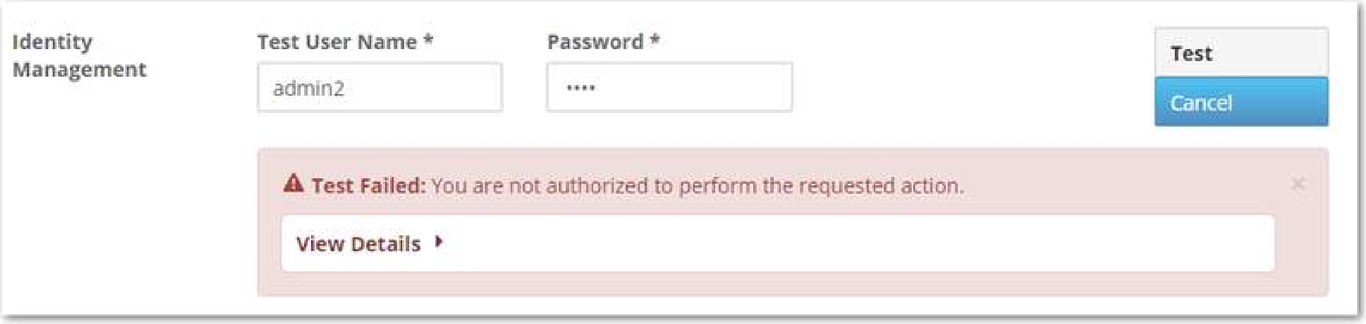
Config Script *

```
{
  "pam": {
    "name": "idsys-pam",
    "description": "This is MyCompany.com PAM configuration"
    "cookieName": "token",
    "tokenPath": "/.TOKEN/" <!-- no comments allowed -->
  }
}
```

Tip
If you click Inherited and lose your script, you can select Revert to restore the last script you saved.

Testing the identity configuration

To test your identity management configuration, click Test, enter a user name and password pair, and click Test.



Identity Management

Test User Name *: admin2

Password *: ••••

Test Failed: You are not authorized to perform the requested action.

Best practice
Remember to test invalid as well as valid user name and password pairs.

Permissions

Permissions are determined by the access control policy, which are the rules that grant (or deny) users and groups the ability to perform specific actions.

> See Setting Permissions.

Tokens

> See Setting Tokens.

Configuring Domains

- Domain Essentials
- Domain Properties

- Permissions
- Tokens

Domain Essentials

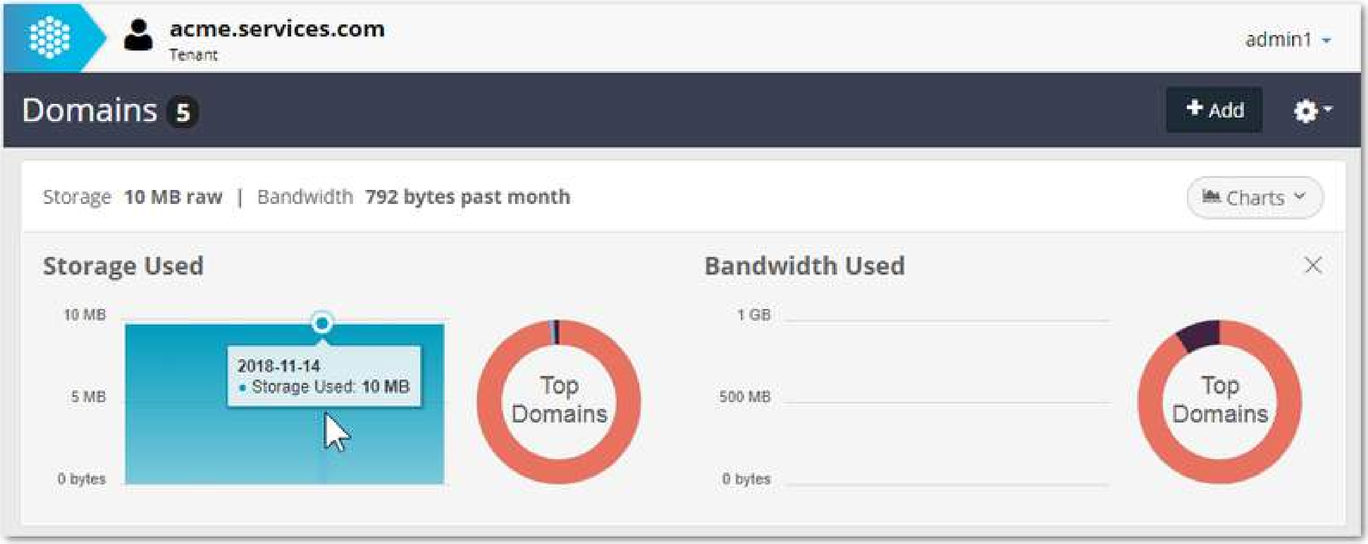
Within a tenant, a domain is the primary entity for dividing and controlling both access and resources. Much like tenants, domains have these essential features:

- Ownership. Each domain owns one or more buckets.
- Access control. Domains can define their own identity management system so that the users and groups within them are separated from those in other domains.
- Delegation. Domain administrators can create and access storage domains and they can delegate management duties for the storage domains that they create.

> See the Naming Rules for Swarm for domains.

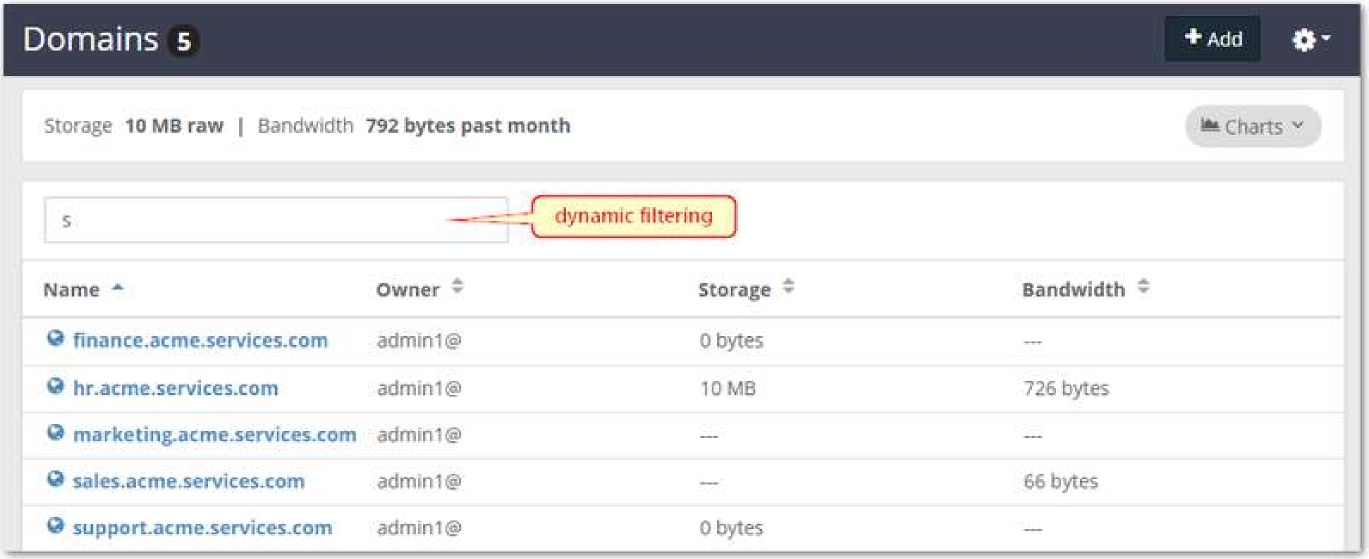
Note
Unnamed objects written directly to the domain are represented by a virtual Content IDs bucket that is part of each domain.

The Storage Used chart displays the current amount of storage used by each domain, inclusive of all versions, replicas and erasure coded segments. The Bandwidth Used chart displays the total bandwidth (both bytes in and bytes out) used by each domain over a rolling 30-day window.



> See Usage Reports.

If you have a large number of domains, you can narrow the listing by entering a string in the Filter box, which filters by Name.

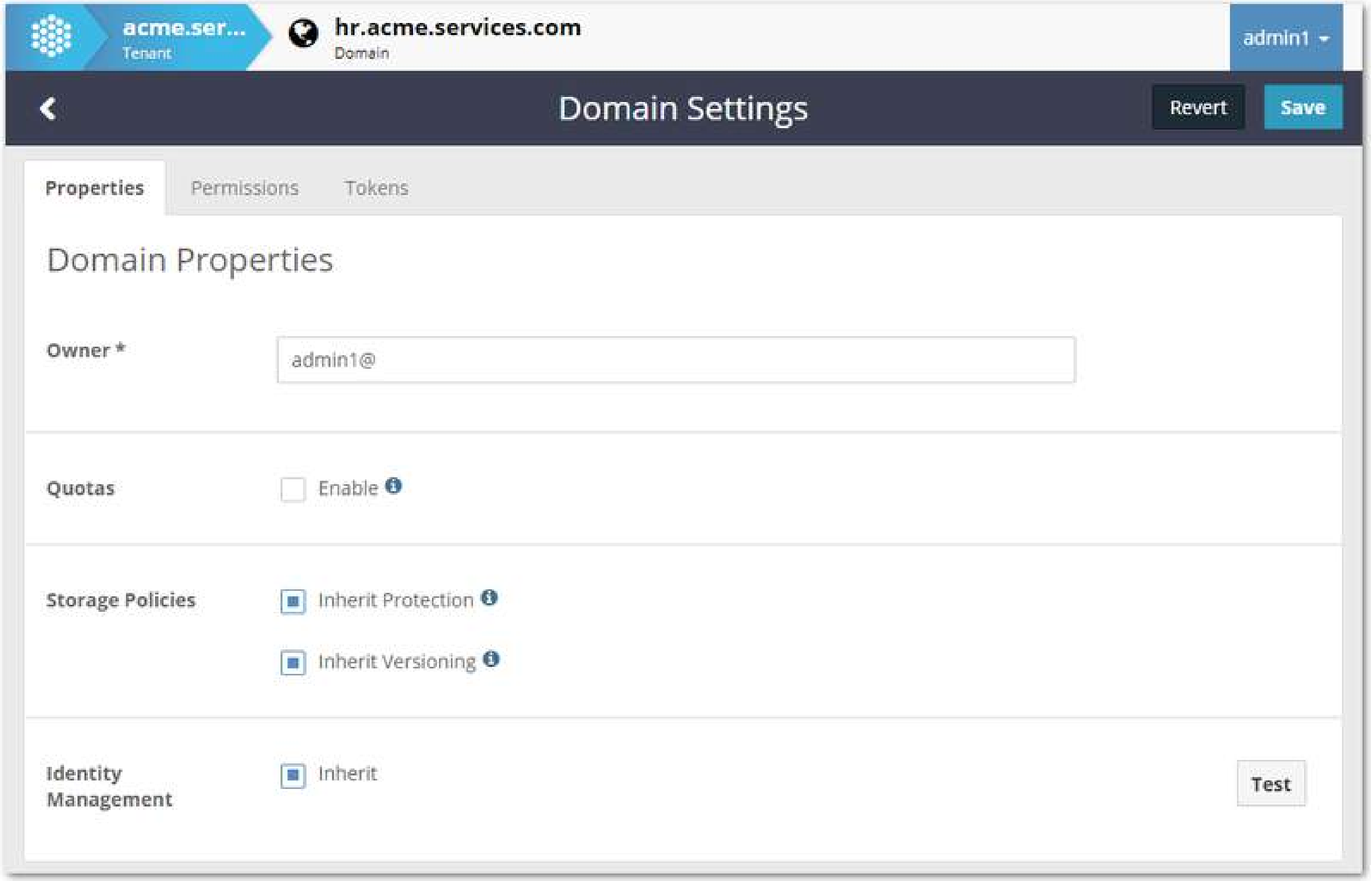


Storage **10 MB raw** | Bandwidth **792 bytes past month**

s

Name	Owner	Storage	Bandwidth
finance.acme.services.com	admin1@	0 bytes	---
hr.acme.services.com	admin1@	10 MB	726 bytes
marketing.acme.services.com	admin1@	---	---
sales.acme.services.com	admin1@	---	66 bytes
support.acme.services.com	admin1@	0 bytes	---

Domain Properties



Domain Properties

Owner * admin1@

Quotas □ Enable

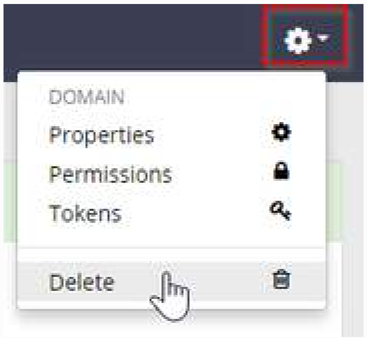
Storage Policies ☑ Inherit Protection

☑ Inherit Versioning

Identity Management ☑ Inherit

Important

The Delete command deletes not only the domain but also all of its buckets and their uploaded contents and any saved collections for the domain. This command cannot be undone, so proceed with caution.



Owner

Every domain must have an owner, who has access to and authority over its entirety. As a root or tenant admin, you can create a domain for another to manage.

> Note
> Ownership defaults to the specific administrator who created the domain, but the owner does not have to be a root or tenant administrator.

In general, change the owner when you are creating a context for someone else to manage. Typically, if you are creating a domain for a customer, you do not want to own or be responsible for managing the data in that domain.

Quotas

Quotas can be set to determine how much storage and/or network bandwidth the domain is permitted to consume.

> See Setting Quotas.

Storage Policies

Storage policies control how this domain's objects are protected (via replication and/or erasure coding) and whether they are versioned. By default, the domain inherits the storage policies that are in force for the cluster.

If you disable inheriting these policies, you can specify custom policies, but be aware that these custom policies are subject to what is allowed and in force in the cluster. If you opt for something that is being overridden by a higher policy, a warning icon and message will alert you to the situation.

> See Setting Storage Policies.

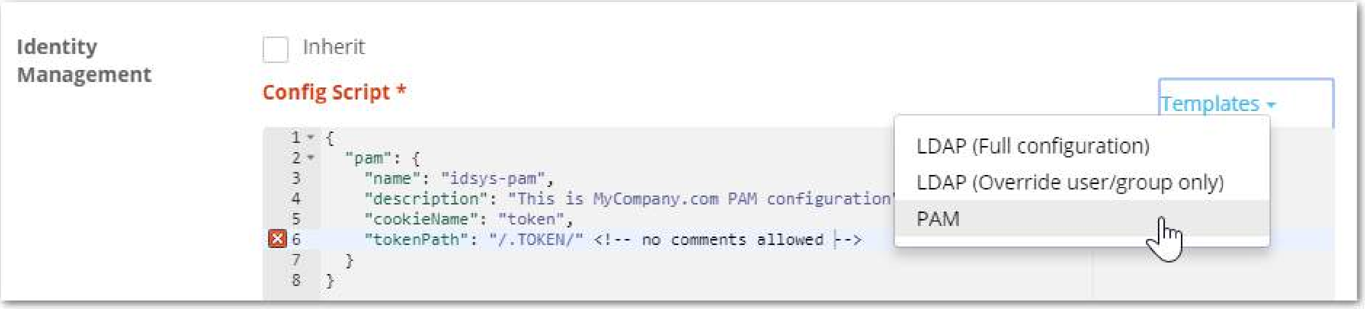
Identity Management Configuration

The IDSYS objects define the identity management systems that control the domain's users, specifically:

- User and group information
- The authentication system

Customizing the configuration script

Every domain inherits the root configuration, but you can disable (uncheck) Inherit and change your configuration. From the Templates drop-down list, you can copy existing definitions to alter (your changes do not affect the originals). Your script is validated in real time:



Identity Management

□ Inherit

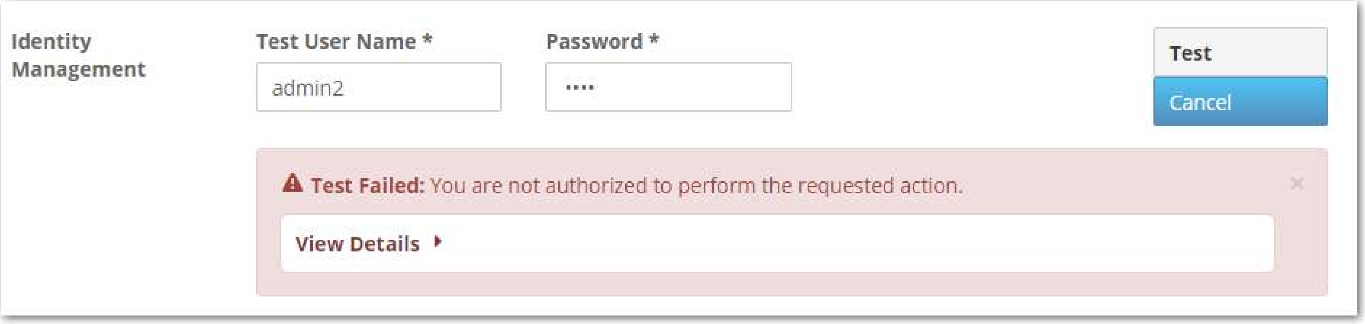
Config Script *

```
{
  "pam": {
    "name": "idsys-pam",
    "description": "This is MyCompany.com PAM configuration"
    "cookieName": "token",
    "tokenPath": "/.TOKEN/" <!-- no comments allowed -->
  }
}
```

Tip
If you click Inherited and lose your script, you can select Revert to restore the last script you saved.

Testing the identity configuration

To test your identity management configuration, click Test, enter a user name and password pair, and click Test.



Identity Management

Test User Name *: admin2

Password *: ••••

⚠ **Test Failed:** You are not authorized to perform the requested action.

Best practice
Remember to test invalid as well as valid user name and password pairs.

Permissions

Permissions are determined by the access control policy, which are the rules that grant (or deny) users and groups the ability to perform specific actions.

> See Setting Permissions.

Tokens

> See Setting Tokens.

Configuring Buckets

- Bucket Essentials
- Naming Buckets
- Bucket Properties
- Permissions

Bucket Essentials

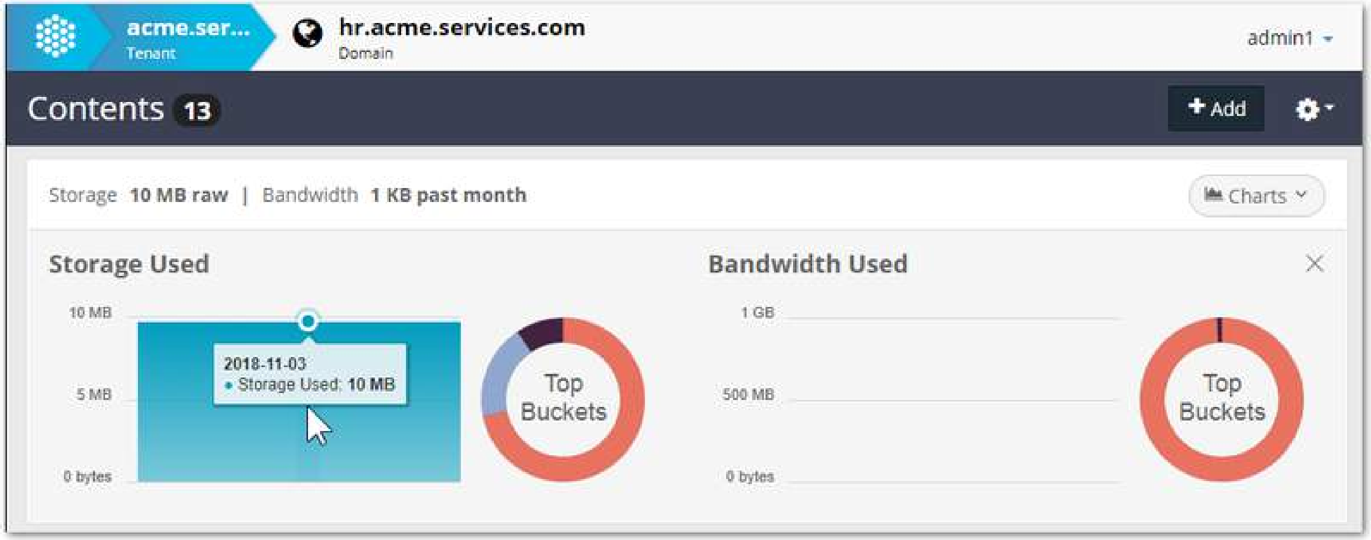
Within a domain, a bucket is the primary entity for managing uploaded content. Buckets have these essential features:

- Ownership. Each bucket has an owner.
- Access control. Buckets can define their own permissions so that the users and groups within them are separated from those in other buckets.
- Content. The bucket itself stores end-user data.

> Note
> Even unnamed objects written directly to the domain are contained in buckets – in this case, the system-controlled, read-only Content IDs bucket that is part of each domain.

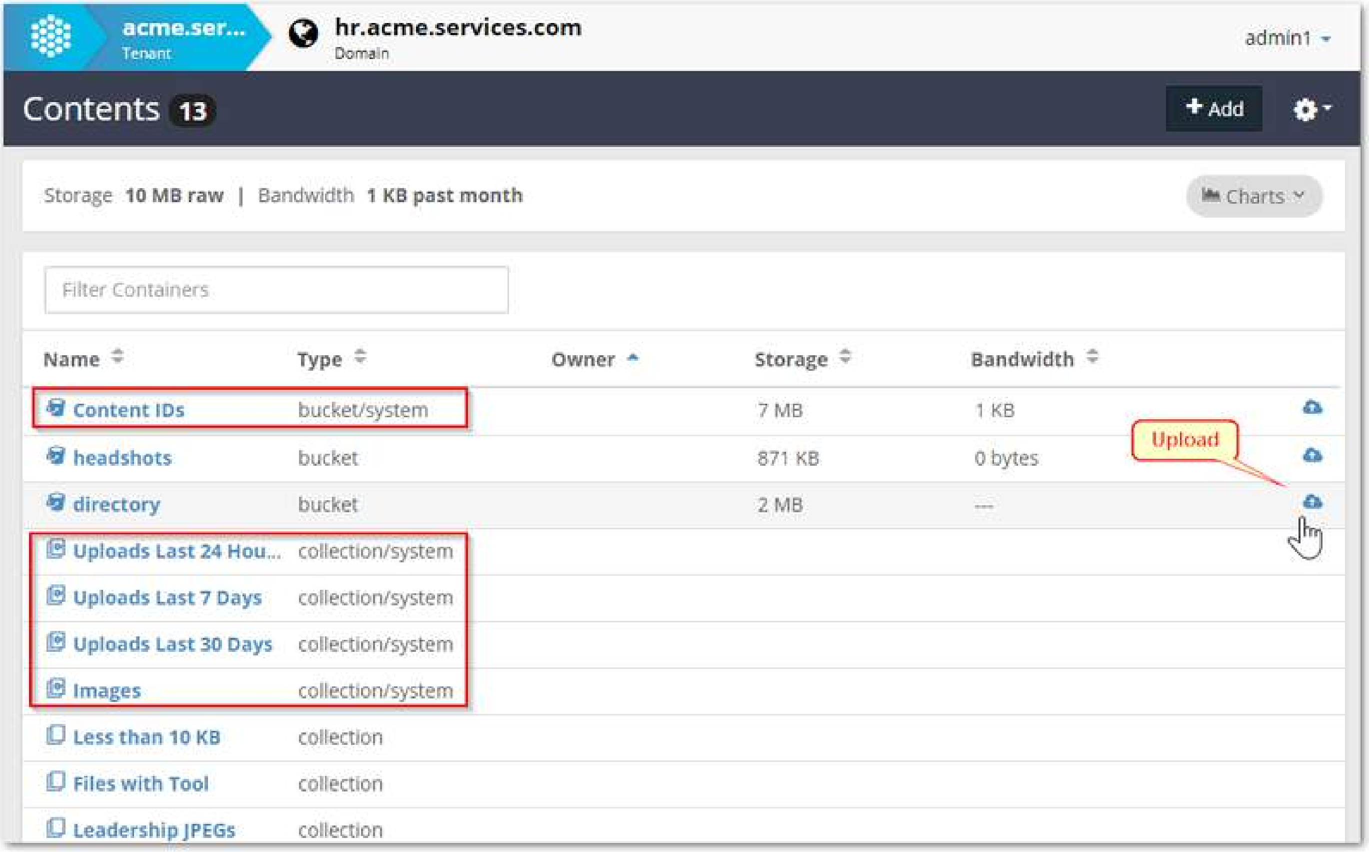
> See the Naming Rules for Swarm for buckets and Bucket Restrictions in Amazon S3.

The Storage column displays the current amount of storage used by each bucket, inclusive of all versions, replicas and erasure coded segments. The Bandwidth column displays the total bandwidth (both bytes in and bytes out) used by each bucket over a rolling 30 day window.



> See Usage Reports.

If you have a large number of buckets and/or collections, you can narrow the listing by entering a string in the Filter box , which filters by Name. You can use the Upload icons to the far right of the listed buckets to initiate uploads from your local file system.



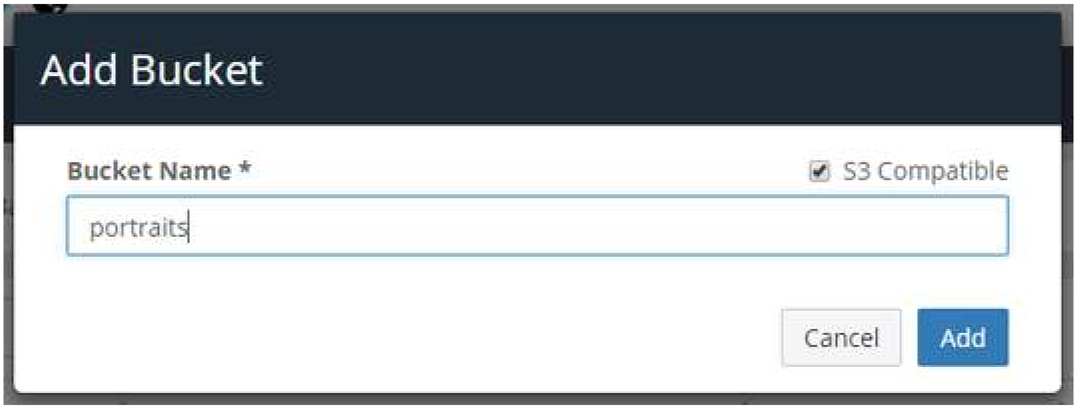
Storage **10 MB raw** | Bandwidth **1 KB past month**

Name	Type	Owner	Storage	Bandwidth
Content IDs	bucket/system		7 MB	1 KB
headshots	bucket		871 KB	0 bytes
directory	bucket		2 MB	---
Uploads Last 24 Hou...	collection/system			
Uploads Last 7 Days	collection/system			
Uploads Last 30 Days	collection/system			
Images	collection/system			
Less than 10 KB	collection			
Files with Tool	collection			
Leadership JPEGs	collection			

Important
Every bucket has a special system-generated bucket for unnamed objects (Content IDs), and every bucket has default search collections, for commonly needed views into the content.

Naming Buckets

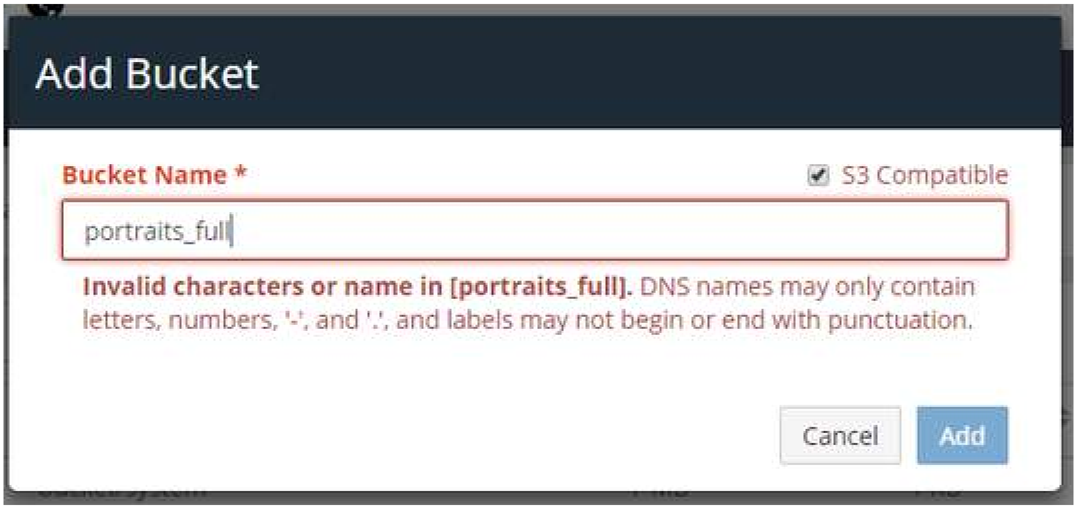
To add a bucket, you must name it immediately. When your name is validated, the Add button becomes active for you to select:



Bucket Name * ☑ S3 Compatible

portraits

S3 compatible — The name that you create needs to use lowercase alphanumeric characters and stay within 3 to 63 characters in length. As you type, the name is validated with dynamic feedback:



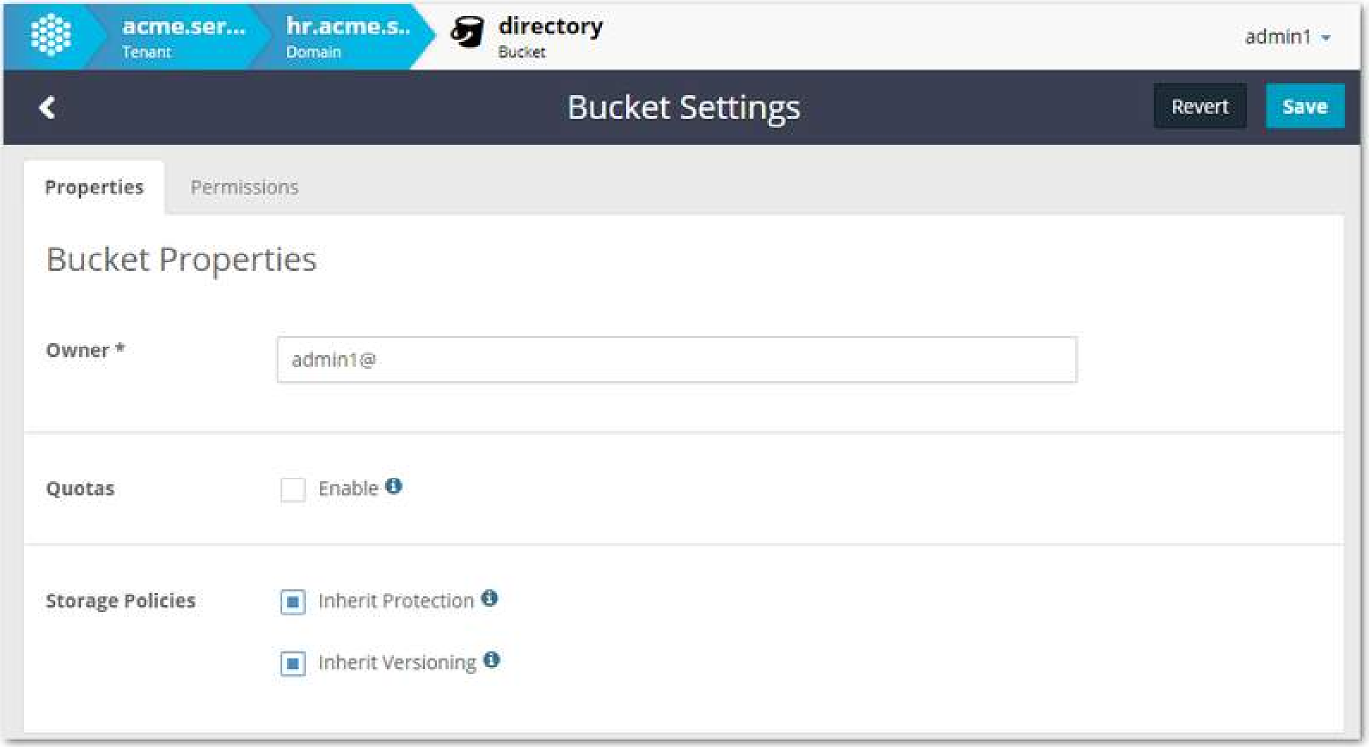
portraits_full

Invalid characters or name in [portraits_full]. DNS names may only contain letters, numbers, '-', and '.', and labels may not begin or end with punctuation.

> See Bucket Restrictions in Amazon S3.

Tip
By following S3 compatibility restrictions in naming, you improve the general compatibility of your bucket names with any future application integrations you may need.

Bucket Properties



Bucket Properties

Owner * admin1@

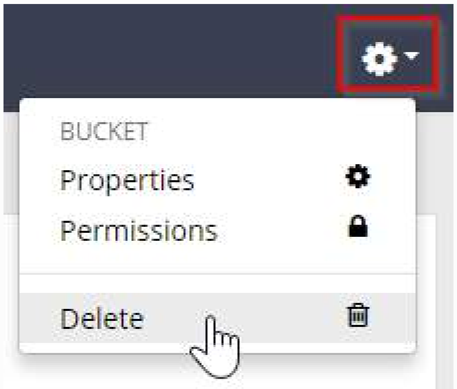
Quotas ☐ Enable

Storage Policies ☑ Inherit Protection ☑ Inherit Versioning

Important

The Delete command deletes not only the bucket but also all of its uploaded contents. This command cannot be undone, so proceed with caution.

Deleted objects may continue to appear in Collection listings for a period of time after they are deleted, but they will no longer be accessible in the cluster.



Owner

Every bucket must have an owner, who will have access to and authority over its entirety.

Note

Ownership defaults to the person who created the bucket, but the owner does not need to be a domain

> administrator.

In general, change the owner when you are creating a context for someone else to manage. Typically, if you are creating a bucket for a customer to use with their application, you will not own or be responsible for managing the data in that bucket.

Storage Policies

Storage policies control how this bucket's objects are protected (via replication and/or erasure coding) and whether they are versioned. By default, the bucket inherits the storage policies that are in force for the cluster and the domain. If you disable inheriting these policies, you can specify custom policies, but be aware that these custom policies are subject to what is allowed and in force in the cluster and the domain. If you opt for something that is being overridden by a higher policy, a warning icon and message will alert you to the situation.

> See Setting Storage Policies.

Permissions

Permissions are determined by the access control policy, which are the rules that grant (or deny) users and groups the ability to perform specific actions.

> See Setting Permissions.

Setting Quotas

- Quota Essentials
- Enabling Quotas
- Defining a Quota
- Quota Effects
- Quota Headers
- API for Quotas
- Example Quota Scenarios

Quota Essentials

The Gateway manages all quota actions; as with Metering, there is no additional service for you to install and monitor. You use the Content Portal to set quotas on a context, just as you set content protection and versioning. A quota policy combines these elements:

1. Scope - where the quota applies: a specific bucket, a domain, or an entire tenant
2. Limits - how much storage and/or network usage is allowed
3. Action - what happens as a result of overage, from simple notification to complete lockout
4. Notification - who to email with information about the quota status change
5. Override - (optional) alternate action as a result of overage with an expiration time

Quota processing relies on periodic queries of Elasticsearch where historical bandwidth and storage metering is maintained. Gateway continually evaluates the metered data against the configured limits on tenants, domains, and buckets in order to determine when limits are exceeded. The action and optional override are used to determine the restrictions that are imposed while a quota remains above its limit.

During quota evaluation, if multiple limits are exceeded, including limits from different scopes within the hierarchy, then the most restrictive action or override will be applied. See Example Quota Scenarios later in this guide for examples of multiple limits interacting within the scope hierarchy.

When an administrator uses an override to force an action that is different from the computed one, an expiration date must be provided. Overrides are designed as a temporary measure to alter the effects of exceeding a quota limit without requiring the action to be changed. Upon reaching the expiration date, the override is automatically invalidated by the Gateway so that an administrator does not need to remember to remove any overrides that they put into place. Once an override expires, the quota's defined action will again apply if the limit is exceeded.

Purpose for Quotas

By default, no quotas exist, which means that all users are entitled to as much network bandwidth and storage space as your storage cluster allows. To create a cloud service that manages and bills tenants, you will want to design quota policies that use metrics to enforce your service level agreements. Scope owners may want to define self-imposed quota limits, such as to put a safety cap on their overall usage in order to control their bill.

Quota Metrics

Quota policies monitor and allow you to limit two classes of metrics: bandwidth usage and/or storage usage:

- Bandwidth: total of network bytes IN plus bytes OUT. Bytes IN refers to data sent from the client application to the front-side interface of the Gateway. Bytes OUT refers to data sent from the Gateway to the client application. These measures include only the content body of the HTTP requests. Bandwidth from replication feeds are included when a feed is routed through the Gateway. Bandwidth from Management API requests are not included.
- Storage in use is one of two types: Raw and Logical. Raw storage refers to the actual amount of disk space used within the cluster based on the replication/EC factor for the objects. Logical storage refers to just the summation of the objects’ Content-Length headers. Only one type is allowed per quota policy.

Cascading Limits

You assign quotas to one or more contexts: tenants, domains, and buckets. Swarm applies quota states in this order: tenants, then domains, then buckets. This means that a bucket’s quota is capped by the limits for its domain and tenant, and a domain’s quota is capped by the limits for its tenant. You are also free to define quota limits that, when combined across a scope, exceed the limits allowed for that scope (such as 12 domains with 1-TB limits being housed in a tenant with a 10-TB limit). This flexibility simplifies your task of quota creation and makes it easier to manage quotas at lower levels.

> Important
>
> Exceeding a metric limit at a higher scope level cascades the overage across all lower scopes. That is, if a tenant’s limits are exceeded, all of its storage domains and all of their buckets are over quota, regardless of their individual metrics. An empty bucket could be over quota because its domain is over quota.

Over-provisioning

When metric limits are displayed in the Content Portal, they show the limits as specified for that scope level. If no quota exists at that scope level, the limits display as if they are unlimited. This means that, for example, the metric limit for a bucket with a quota would display that bucket’s metric limits even if they exceed the domain’s metric limits or the tenant's metric limits. This allows administrators to over-provision storage for lower scopes. The Content Portal displays those limits in the lower scopes as if they will be possible before the limits at the higher scopes are exceeded.

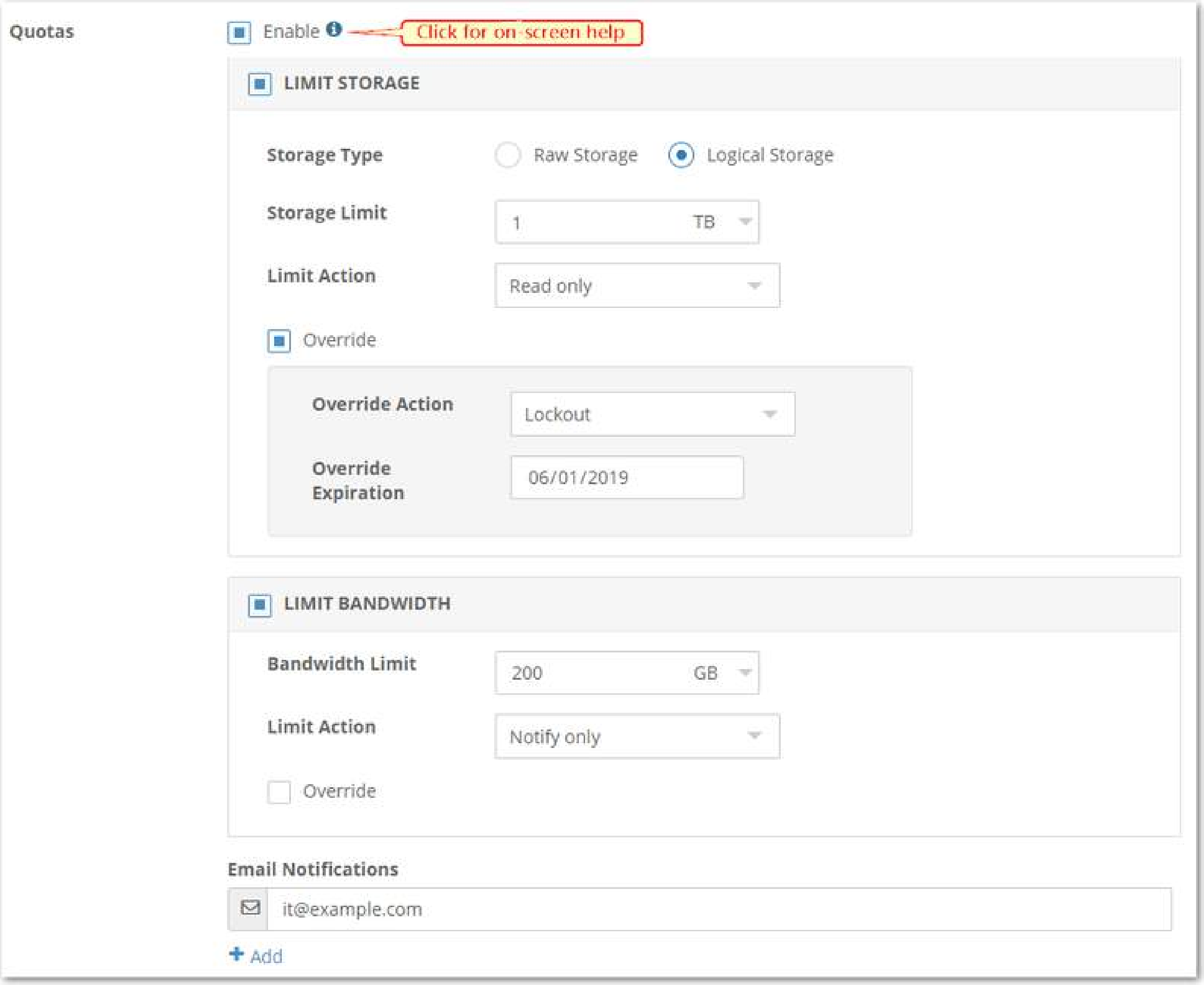
Enabling Quotas

By default, the Quotas feature is disabled globally, and it must first be enabled through Gateway. The Gateway configuration has a `[quota]` section for enabling, controlling, and customizing your notifications for quotas.

> See Gateway Configuration.

Defining a Quota

To define a quota, you first go to the Properties (gear icon) of the tenant, domain, or bucket that you want to control. Click Enable to view the policy settings:



Quotas ☑ Enable ℹ

Click for on-screen help

☑ LIMIT STORAGE

Storage Type ○ Raw Storage ● Logical Storage

Storage Limit: 1 TB

Limit Action: Read only

☑ Override

Override Action: Lockout

Override Expiration: 06/01/2019

☑ LIMIT BANDWIDTH

Bandwidth Limit: 200 GB

Limit Action: Notify only

☐ Override

Email Notifications

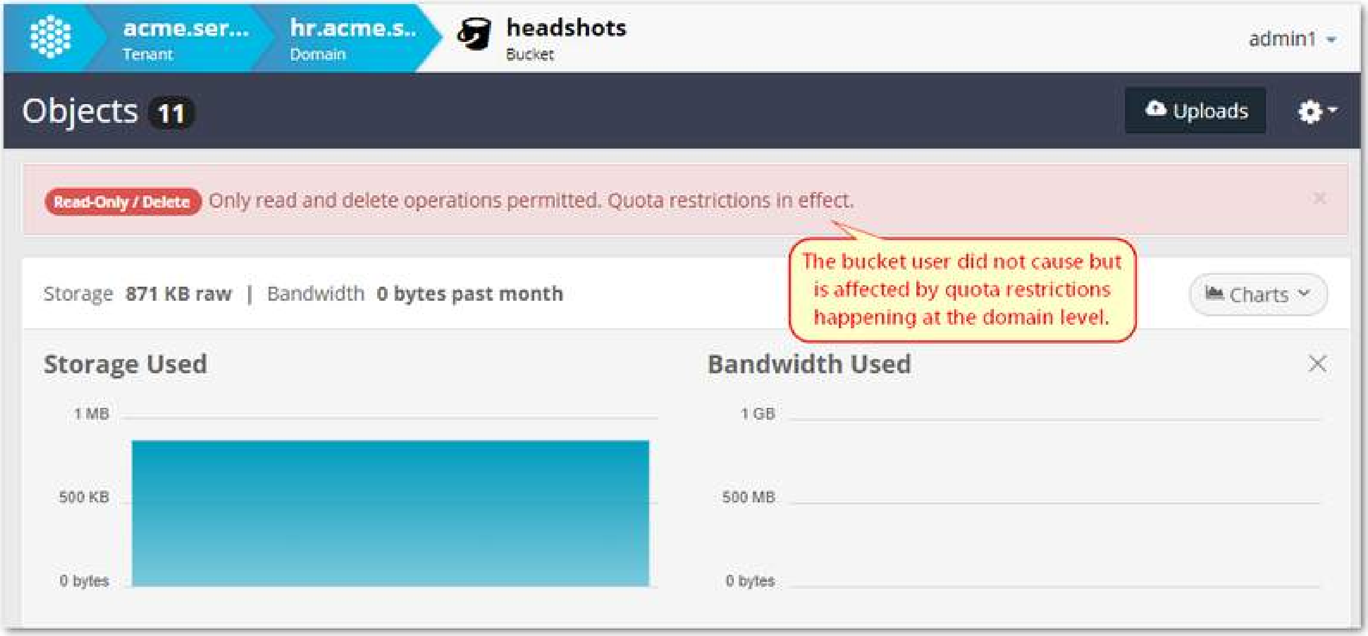
✉ it@example.com

+ Add

Enable	The feature must be deliberately enabled for this scope, after which all required values must be entered in order to Save.
Storage Type	Choose to monitor storage usage by • Raw Storage (the total footprint on disk of all objects, replicas, segments, manifests) • Logical Storage (the sum of the uploaded content and any versions).
Storage/Bandwidth Limit	Limits are set in units of MB, GB, TB, or PB. Note Bandwidth resets automatically based upon a calendar month: the first day of the month at 00:00:00 UTC. When a metric falls back within the allowed limit or resets for the new month, the status automatically changes to OK.
Actions	Consequences for exceeding a quota, from least restrictive to most: 1. Notify only: Sends notification but does not restrict any operations 2. Read/Delete only: Blocks operations that might increase storage (no writes, updates) 3. Read only: Blocks operations that change or add content (no writes, updates, deletes) 4. Lockout: Blocks all operations
Override	Sets an alternate Action with an Expiration date as a temporary policy override, such as to grant a grace period for the customer to resolve the overage. Overrides may impose a more restrictive Action than the policy, such as might be used if a tenant is delinquent in their payments.
Email Notifications	Lists recipients of email notifications, who will be notified when a limit is exceeded or is no longer exceeded. Every time an overage begins or ends, the Gateway sends an email to every address defined within the quota policy's notification email list. These notifications contain the scope, the metric’s name and limit, the time of detection, and the resulting quota state that now applies to the scope. The scope is identified as the tenant name, the domain name, or the domain + bucket name. Notifications are specific to the scope where the overage occurs. For example, if a tenant quota is exceeded, notifications do not cascade down to the people in the domain and bucket quotas within that tenant. That is, limits cascade to lower levels, but notifications do not.

Quota Effects

When a metric limit has been exceeded by a tenant, domain, or bucket, the Content Portal displays an alert message about the overage on its Contents page and gives the scope in which it has occurred (the current level or higher). Note that the overage may not be the fault of the current context. For example, if a tenant has exceeded its storage quota, a user of a domain or bucket within that tenant would see an alert message that quota levels are exceeded, even if their domain or bucket is using none of its allowed usage:



Only read and delete operations permitted. Quota restrictions in effect.

Storage 871 KB raw | Bandwidth 0 bytes past month

The bucket user did not cause but is affected by quota restrictions happening at the domain level.

> Note
> Quota metrics are measured periodically, so there can be a lag between when a metric limit is actually exceeded and when the overage is detected. A similar lag can exist going the other direction, as an overage ends. While the typical lag is 5 to 15 minutes in a typical deployment, it is never longer than 60 minutes.

HTTP response for overage - When a storage action fails due to an exceeded quota, the HTTP response indicates where the overage occurred, the metric that was exceeded, and the current quota state. For example, if a domain's storage limit is exceeded while a bucket's storage limit is not exceeded and the quota state is "read only," a write operation to the bucket returns an error that says that the domain's storage quota is exceeded.

Quota Headers

You define quota limits on context objects: tenants, domains, and buckets. Quota management makes use of metadata headers that are stored with these context objects.

In the header names, {M} is one of these usage metrics:

- `bandwidth` (network bandwidth inbound and outbound)
- `rawstorage` (storage in use as the total footprint on disk)
- `storage` (storage in use as the total logical size only)

Header	Value	Description

x-caringo-meta-quota-{M}-limit	= {state} ; {limit} • limit = target for this particular metric • state = “notify”, “nowrite”, “read”, “lock”	The configured quota limit and the Action (consequence) when the limit is exceeded.
x-caringo-meta-quota-{M}-current	= {state} ; {value} ; {timestamp} • value = current for this particular metric • state = “ok”, “notify”, “nowrite”, “read”, “lock” • timestamp = ISO-8601 timestamp of last update	Last measured value of storage and bandwidth metrics as queried from Elasticsearch. This header is computed by Gateway and should not be set externally.
x-caringo-meta-quota-{M}-override	= {state} ; {user} ; [{deadline}] • deadline = ISO-8601 timestamp of expiration • state = “ok”, “notify”, “nowrite”, “read”, “lock” • user = who set the override	Optional. The state and expiration time for the temporary override.
x-caringo-meta-quota-email	= {addresses} • addresses = comma-separated list of email addresses	One or more email addresses to notify about quota state changes Notifications get sent whenever the current state of a metric changes at that context level (tenant, domain, or bucket). The message text is similar to the error messages returned to storage operations that are blocked. The Gateway Configuration has a [quota] section where you customize the email service, the sender, and content of the email notifications for quotas.

API for Quotas

You can set and clear quotas and check on quota states using the Content Management API.

> Note
>
> On Management API context listings, additional headers that are related to quotas may appear in the listing. These may be ignored.

> See Methods for Quotas.

Example Quota Scenarios

Here are two detailed scenarios for quota enforcement actions and how they apply within the scope hierarchy of tenants, domains, and buckets.

```
Tenant alpha: storage quota 1.0PB, limit action "read/delete only"
    Domain alpha-one: no quota
    Bucket mike: bandwidth quota 100TB, limit action "locked"

    Domain alpha-two: no quota
        Bucket november: no quota
```

During use, the sum of the storage in the two domains for tenant alpha grow past 1.0 PB. Tenant alpha's storage quota has now been exceeded and enters a “read/delete only” state. This restriction also propagates down and applies to domains alpha-one and alpha-two and all of their buckets. No one is allowed to add more content -- only read and delete operations.

Later, as reading operations continue on bucket mike within domain alpha-one, the bandwidth total passes 100 TB and bucket mike's quota is exceeded and enters a “locked” state.

At this point, no activity is allowed for bucket mike since it is “locked” while the other domains and buckets within tenant alpha remain in a “read/delete only” state.

After the end of the month is reached and bucket mike's bandwidth is reset to zero, bucket mike is longer exceeding the bandwidth quota. However, since the tenant is still in a “read/delete only” state, bucket mike returns to the inherited state of “read/delete only.”

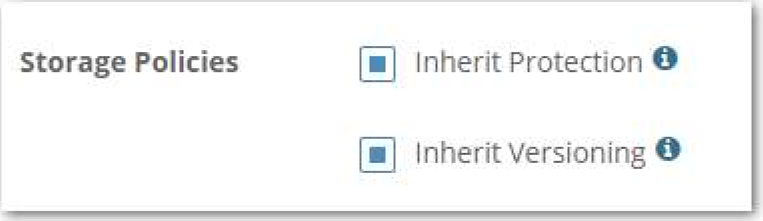
```
Tenant bravo: bandwidth quota 500GB, limit action "locked"
    Domain bravo-three: storage quota 2.0PB, limit action "read only"
        Bucket oscar: no quota

    Domain bravo-four: no quota
        Bucket papa: bandwidth quota 250GB, limit action "notify only"
```

During use, domain bravo-three exceeds its storage quota of 2.0 PB and enters a “read only” state. Because of inheritance, this also means that bucket oscar is now read-only. Later, bucket papa within domain bravo-four exceeds its bandwidth limit of 250 GB and enters a “notify only” state. Since all actions are still allowed, bucket papa can continue to use additional bandwidth. As this continues, tenant bravo eventually exceeds its bandwidth total of 500 GB and enters and “locked” state. Due to inheritance, this locked state now applies to domains bravo-three and bravo-four and to their buckets. Thus, bucket oscar and papa are now locked because of the tenant quota limit.

A few frantic phone calls after this lockout happens, the administrator agrees to override tenant bravo's bandwidth quota with “notify only” for the remainder of the month. Thus, tenant bravo's effective state is now “notify only” and all storage operations are again allowed for the lower scopes except for bucket oscar. Bucket oscar remains in a "read only" state because it is still over its storage limit and no override exists for its quota. Bucket papa remains in "notify only" state, which is the same coming from tenant bravo, thus, storage operations are restored even though bucket papa remains over their limit.

Setting Storage Policies

When you edit the Properties of a domain or bucket, you have the option to specify storage policies that you want to apply to the objects that it contains:



Storage Policies ■ Inherit Protection ℹ

■ Inherit Versioning ℹ

By default, the domain or bucket inherits the protection setting that are in force above it (cluster, domain).

- To see the options for a storage policy, deselect the Inherit checkbox, which expands the policy section.
- For guidance about the policy options, click the information icon, which toggles the help text.

Protection

Swarm allows you to flexibly determine the type and level of content protection that best fits your storage needs. In Swarm storage, objects can be replicated and/or erasure-coded, with objects of both types co-existing in the same cluster.

> Tip
> Erasure coding helps cost-effectively scale clusters that have many nodes and larger objects, while replication is better for smaller clusters and with smaller objects.

These settings let you choose replication and erasure-coding protection policies for the objects in this immediate context, subject to overrides by higher-level (cluster or domain) settings.

Replication

Replication protection requires the cluster to keep a specified number of copies (replicas) of each object.

- Default Replicas: Accept the inherited number or enter how many replicas you want (subject to existing min and max values and query args).
- Anchored: Select to override any lower-level policies.

> For details about replication in Swarm, see Replication.

Erasure Coding

Erasure coding protection divides very large objects into multiple data (k) and parity (p) segments for distribution across k+p nodes.

- Enabled: Select to allow erasure coding at this level and below (subject to higher-level policies).
- EC Size Threshold (MB): (not settable) Reports the object size that will trigger erasure coding rather than replication.
- Default Encoding: Accept the inherited encoding, or enter data (k) and parity (p) values such as these examples:
  - 5 : 2 (1.4x footprint) - protection for 2 simultaneous disk failures.
  - 9 : 6 (1.7x footprint) - protection for 6 simultaneous disk failures and 1 subcluster failure in clusters of 3 or more subclusters.
- Anchored: Select to override any lower-level policies.

> For details about erasure coding in Swarm, see Erasure Coding EC.

Versioning

Swarm supports object-level versioning, which is a powerful content protection option that tracks, secures, and provides access to historical versions of objects, even after they are deleted. With versioning, your applications can read, list, revert, and purge prior versions as well as restore objects that were deleted by mistake.

Best practices

- Plan for higher disk utilization with versioning: each update to a versioned object adds a new object to the cluster (one object updated twice results in three objects stored).
- Where possible, make use of lifepoints to control the lifetime – and thus the cost of storing – multiple versions of your objects.
- For optimal resource management, limit versioning to the specific domains and/or buckets for which it is needed.

> For details about versioning in Swarm, see Object Versioning.

The versioning state of the immediate context applies to every object in that context, without exception. Each domain and bucket has one of these versioning states:

disabled	(default) No versioning exists, so no versions are created. This state is the normal behavior of Swarm.	If you change from enabled to disabled in the domain or bucket, the Health Processor will clean up all of the residual prior versions, regaining that space. This feature (changing to disabled) is not available in Amazon S3.

suspended	No new versions are accumulated but old versions are retained. This is a hybrid between enabled and disabled that preserves history.	If you re-enable versioning from this state, the chain of versions resumes from where it stopped.
enabled	New versions are accumulated as they are created, starting with any version that exists at the time versioning becomes enabled for the object.	This is the only parent state from which domains and buckets can enforce a versioning policy that differs from its parent.
required	Domains only. Requires each of its buckets to have versioning enabled.	Use this to prevent bucket owners from disabling your versioning policy.

Important

By default, versioning is disabled at the cluster and every other level. Versioning must be enabled for the cluster through the Swarm configuration setting `policy.versioning`. Cluster-level values are `disallowed` (de fault), `suspended`, and `allowed`. In a cluster that has versioning allowed, every newly created domain and bucket starts with an unspecified state, so object versioning is disabled until you enable it there explicitly.

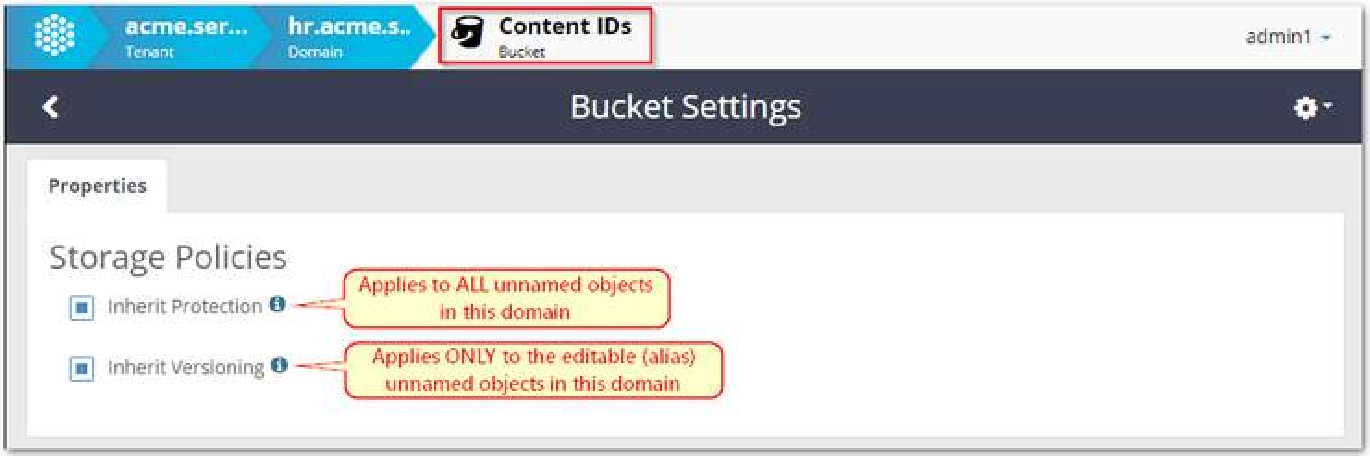
Reminder

The versioning option you select depends on what's permitted in the given context. If your domain has versioning enabled, then versioning would occur in a cluster that allows it but not in one that disallows it, such as a remote replication cluster.

Policies for Unnamed Objects

All named objects are controlled by the policies on their buckets, but unnamed objects are handled separately for each domain.

To view and set the storage policy for unnamed objects in the domain, view All Buckets and open the special Content IDs bucket, which is the system-controlled container for all unnamed objects in the domain. Open its settings (cog icon) and specify the policy to apply specifically to the unnamed objects in the domain:



☑ Inherit Protection — Applies to ALL unnamed objects in this domain

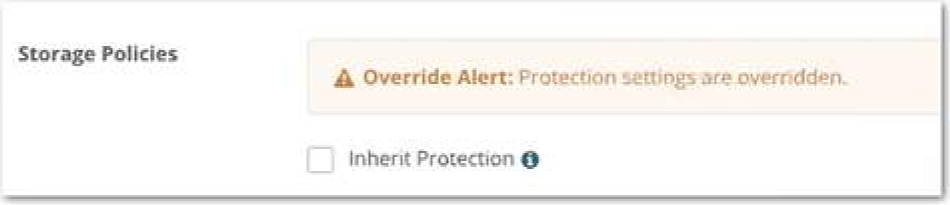
☑ Inherit Versioning — Applies ONLY to the editable (alias) unnamed objects in this domain

Because immutable objects cannot be updated, they cannot be versioned.

> Reminder
> The versioning policy on the Content IDs bucket is special and independent of the policies in force on the domain's buckets.

Override Alerts

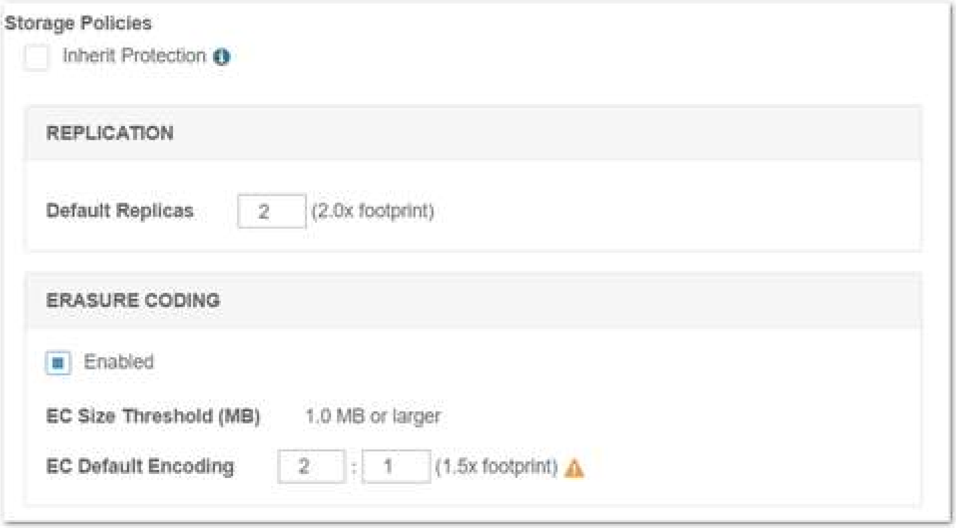
Because the policies you define are subject to override by policies at higher levels (such as the cluster settings), alert icons and messages will inform you if the policy you specified is blocked from going into effect.



⚠ **Override Alert:** Protection settings are overridden.

□ Inherit Protection

> Important
> Your policy settings are still valid, even if they do not go into effect immediately. If the higher-level policy changes at a future time or if your content is replicated to a cluster with different policies, your policy request may be implemented.



Storage Policies

□ Inherit Protection

REPLICATION

Default Replicas 2 (2.0x footprint)

ERASURE CODING

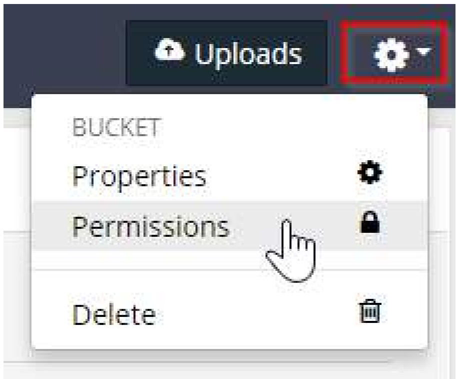
☑ Enabled

EC Size Threshold (MB) 1.0 MB or larger

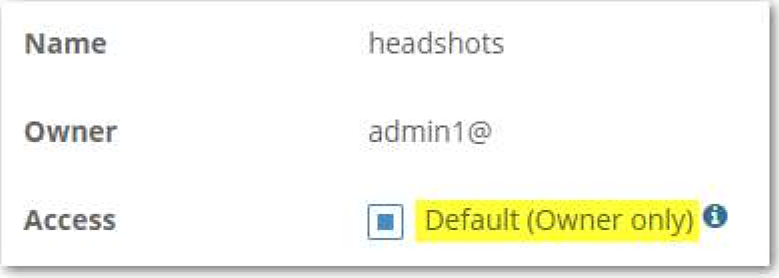
EC Default Encoding 2 : 1 (1.5x footprint)

Setting Permissions

Permissions are determined by the active access control policy, which is a list of rules that grant or deny users and groups the ability to perform specific actions.



Default (Owner only) access applies automatically, referring to the owner of the current tenant, domain, or bucket. In the absence of an access control policy, only the owner will have access unless a parent scope specifically grants additional permissions to other users and groups.



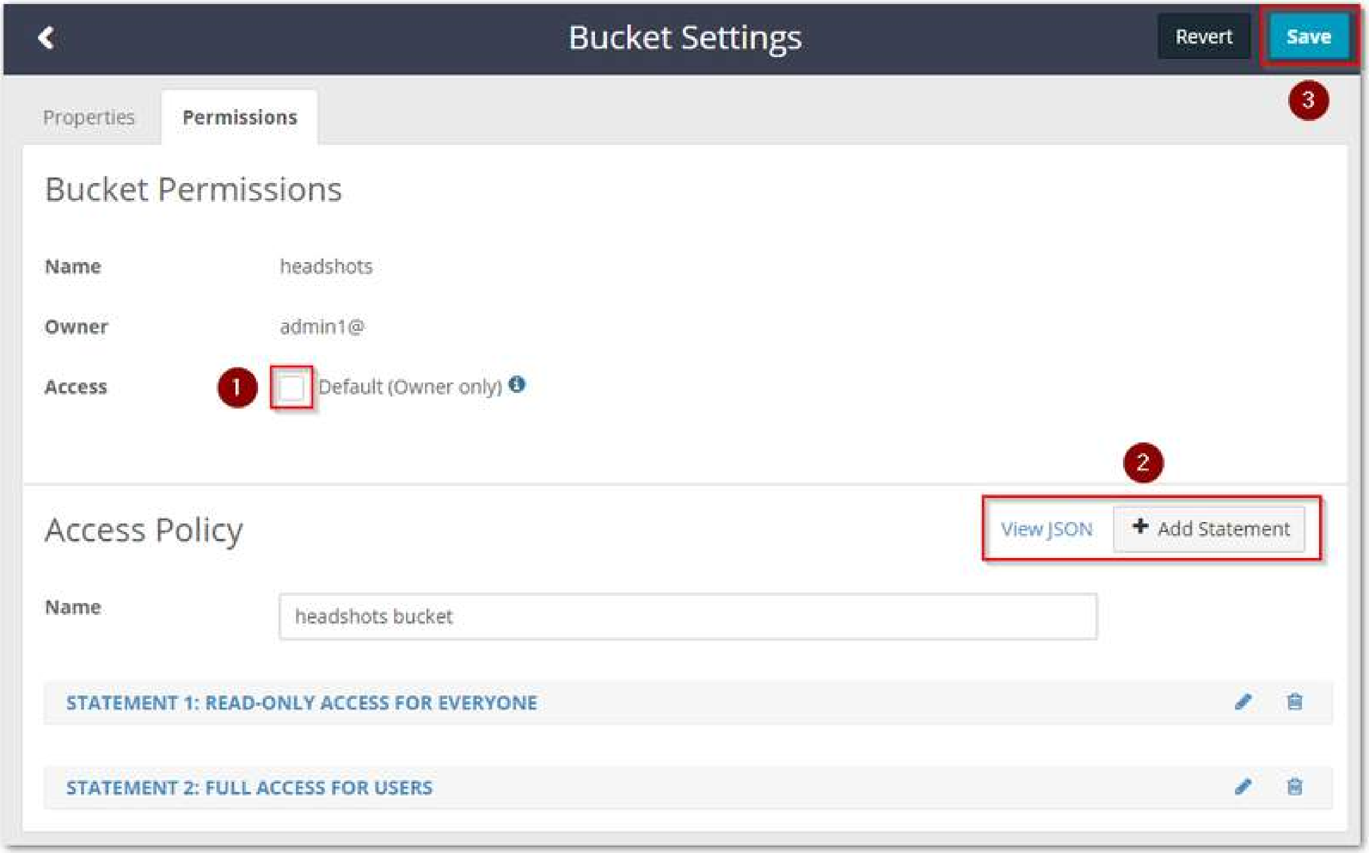
Name headshots

Owner admin1@

Access ☑ Default (Owner only)

> For details about the usage and components of policies, see Gateway Access Control Policies.

When you uncheck Default (Owner only), an interactive policy editor expands so that you can create a policy for the current tenant, domain, or bucket. (v9.4)



Bucket Permissions

Name headshots

Owner admin1@

Access □ Default (Owner only)

Access Policy

Name headshots bucket

STATEMENT 1: READ-ONLY ACCESS FOR EVERYONE

STATEMENT 2: FULL ACCESS FOR USERS

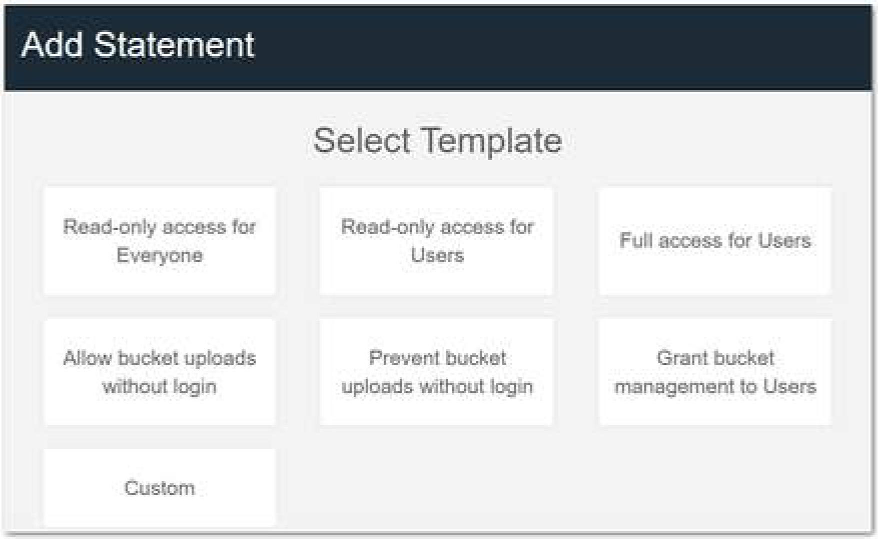
The editor includes templates for adding the most commonly needed policies (such as public read-only and authorized user full access) as well as options for designing granular access for users and groups. Safeguards help protect you from unintended consequences, such as denying access to All Authorized Users, which would have the effect of locking out the Owner as well.

Note these behaviors and cautions:

<table>
<tr>
<td>Add statement</td>
<td>

From the + Add Statement dialog, you can copy existing definitions to alter (your changes do not affect th that is closest to your desired policy, and then edit it for your needs.

> Tip
> Remember to rename the default statement name to describe its new effect: click the Edit (pencil)



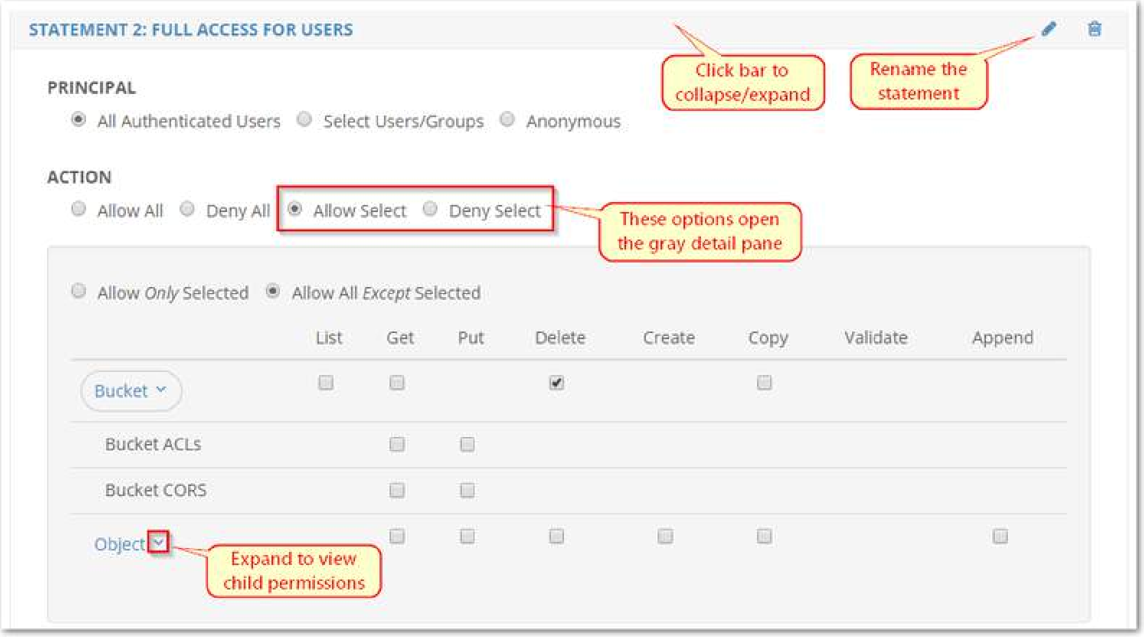
Add Statement

Select Template

Read-only access for Everyone
Read-only access for Users
Full access for Users
Allow bucket uploads without login
Prevent bucket uploads without login
Grant bucket management to Users
Custom

</td>
</tr>
<tr>
<td>View statement</td>
<td>

The title bar of each statement is a toggle: click it to expand and hide the statement settings



STATEMENT 2: FULL ACCESS FOR USERS

PRINCIPAL

● All Authenticated Users ○ Select Users/Groups ○ Anonymous

ACTION

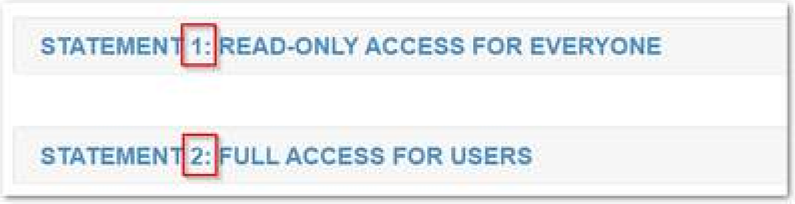
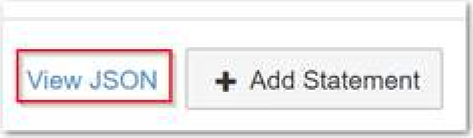
○ Allow All ○ Deny All ● Allow Select ○ Deny Select

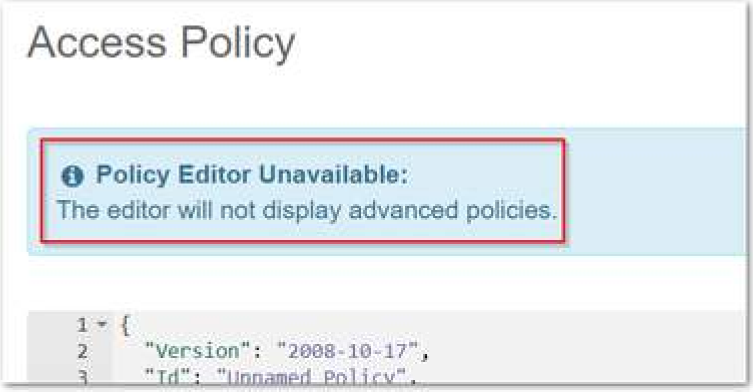
○ Allow *Only* Selected ● Allow All *Except* Selected

<table>
<tr><th></th><th>List</th><th>Get</th><th>Put</th><th>Delete</th><th>Create</th><th>Copy</th><th>Validate</th><th>Append</th></tr>
<tr><td>Bucket</td><td>□</td><td>□</td><td></td><td>☑</td><td></td><td>□</td><td></td><td></td></tr>
<tr><td>Bucket ACLs</td><td></td><td>□</td><td>□</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Bucket CORS</td><td></td><td>□</td><td>□</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Object</td><td></td><td>□</td><td>□</td><td>□</td><td>□</td><td>□</td><td></td><td>□</td></tr>
</table>

Click bar to collapse/expand
Rename the statement
These options open the gray detail pane
Expand to view child permissions

</td>
</tr>
<tr>
<td>Undo edits</td>
<td>To undo any unsaved policy changes that you made, select Revert. This clears any changes that are pendi</td>
</tr>
</table>

Delete policy	To remove a single statement, click the Delete (trashcan) icon in its title bar. The change will take effect v To remove the entire existing access policy, re-enable Default (Owner only) and select Save. Use caution,
Statement counter	The counter prefix that the editor adds to statement names ensures that each statement name is unique; JSON editing view.  STATEMENT 1: READ-ONLY ACCESS FOR EVERYONE STATEMENT 2: FULL ACCESS FOR USERS
View JSON	To view (and optionally edit) the underlying JSON, select View JSON; select Hide JSON to return to the int changes prohibit its use (see next).  Important In the JSON view, `"Version"` specifies the version of the AWS policy language, not the policy's con to the current (`2012-10-17`) or prior (`2008-10-17`) language version, or else JSON validation errors If you need to find the date and user responsible for the last policy update, read the modification tir as standard metadata on the relevant policy.json object.

Advanced policies	The interactive editor will not open for advanced policies, which are those that involve these complexities
	● Resource — Anything other than asterisk (*), meaning all
	● Principal — Any of these:
	  ● Is AWS
	  ● Has any conditions (such as match criteria) with or without child properties
	Advanced policies can only be viewed and edited through the JSON view. If your existing policy has any o interactive editor will be disabled and you will edit the JSON directly:
	
	Access Policy
	Policy Editor Unavailable:
	The editor will not display advanced policies.
	`{`
	`    "Version": "2008-10-17",`
	`    "Id": "Unnamed Policy",`

Prefixes	Prefixes are deprecated and hurt performance, and they are ignored by the policy evaluation. The interacti remove them, which ensures that policies work as expected. This includes the following: `ldap, pam, ar`

Setting Tokens

- Token Essentials
- Accessing Tokens
- Creating Tokens
- Managing Tokens

Token Essentials

In addition to HTTP Basic authentication, Gateway lets you configure token-based authentication. Token-based authentication works in two steps:

1. Request a token, by using HTTP Basic authentication to perform a one-time authentication within the Management API or to a special URI path in the Storage API.
2. Submit this token on all subsequent requests as proof of the user's credentials.

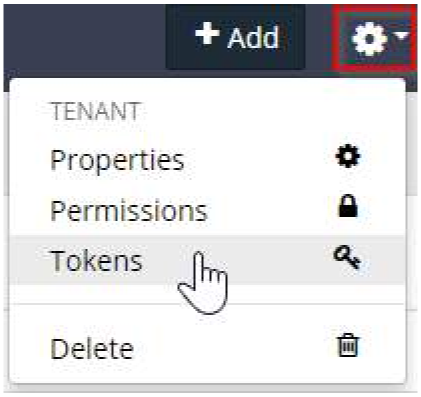
Tokens have these characteristics:

- Ownership. They are always owned by the user that creates them, except for tokens created by the token administrator.
- Expiration. They expire at a fixed time after creation; default is 24 hours.
- S3 key. They may contain an optional secret access key for use with the S3 protocol.
- Deletion. Both the owner and the token administrator can list and delete the owner's active tokens.

> See Token-Based Authentication.

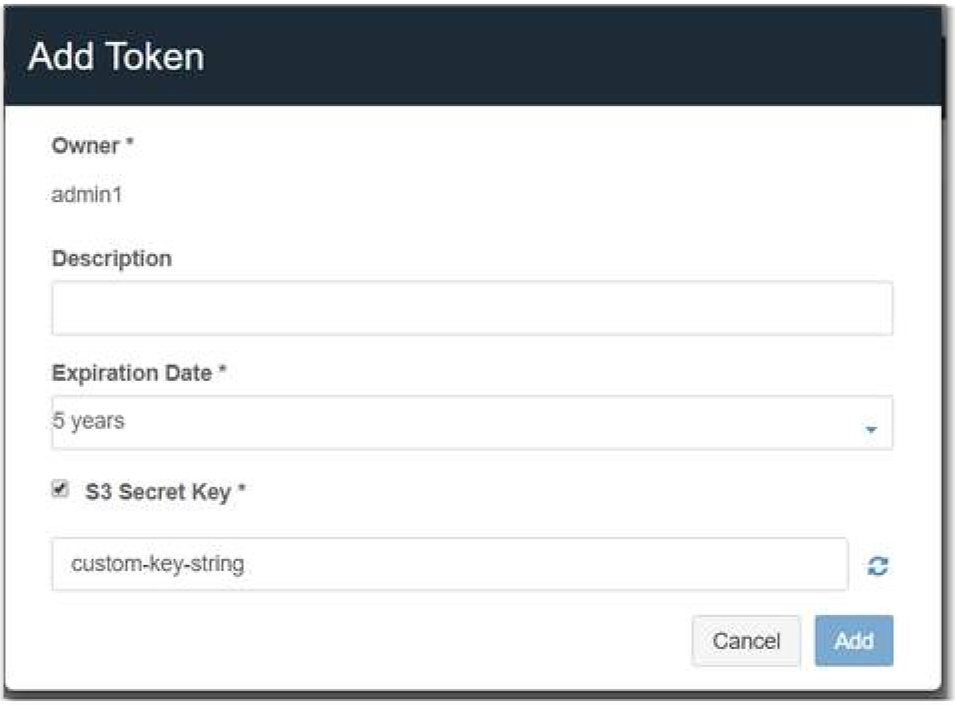
Accessing Tokens

You can access tokens under the gear icon, which appears in the title bar of all tenants and domains (not buckets):



Creating Tokens

When you create a token manually (for the current tenant or domain), you can override the default owner and expiration date, as well as choose to enable the S3 Secret Key:



Add Token

Owner *

admin1

Description

Expiration Date *

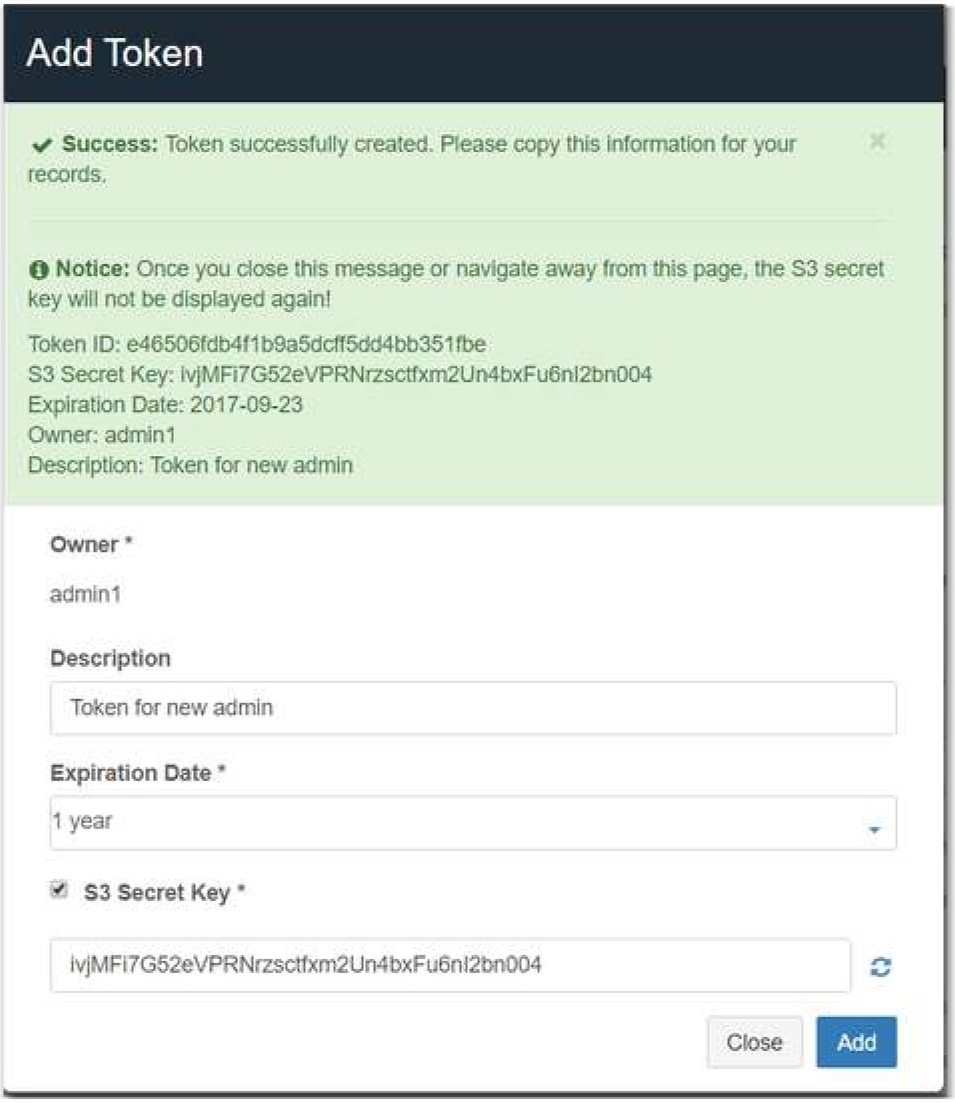
5 years

☑ S3 Secret Key *

custom-key-string

Important

If you chose to enable an S3 Secret Key for the token, you must copy it from the Success message before closing it: for security reasons, the S3 Secret Key is not displayed in the Content UI after this point.



Add Token

✓ **Success:** Token successfully created. Please copy this information for your records.

Notice: Once you close this message or navigate away from this page, the S3 secret key will not be displayed again!

Token ID: e46506fdb4f1b9a5dcff5dd4bb351fbe
S3 Secret Key: ivjMFi7G52eVPRNrzsctfxm2Un4bxFu6nl2bn004
Expiration Date: 2017-09-23
Owner: admin1
Description: Token for new admin

Owner *
admin1

Description
Token for new admin

Expiration Date *
1 year

☑ S3 Secret Key *
ivjMFi7G52eVPRNrzsctfxm2Un4bxFu6nl2bn004

Best practice
If someone loses the S3 Secret Key, delete the token and create a new one so that security is not compromised.

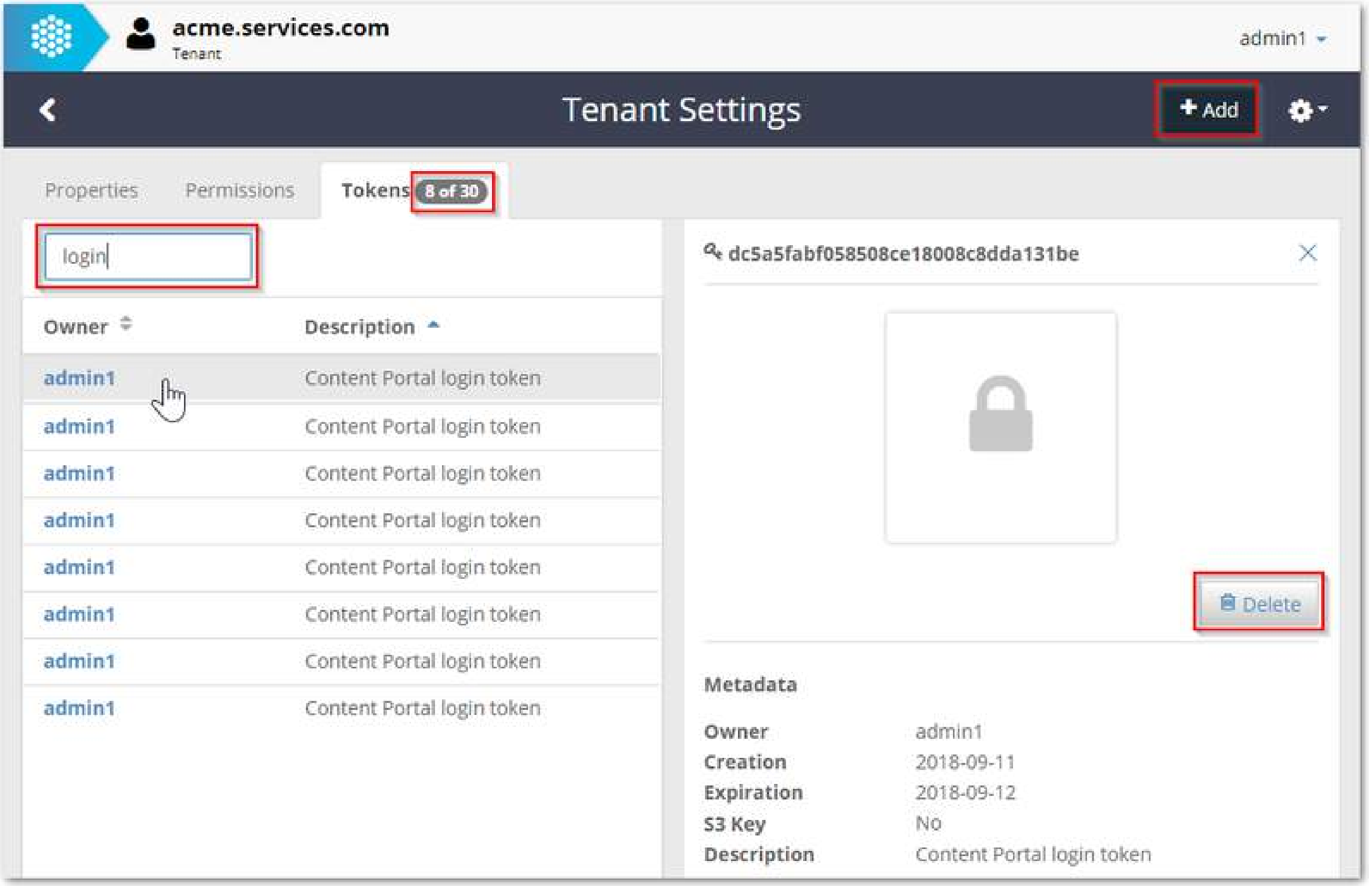
> See Integrating S3 Applications.

Managing Tokens

Be aware that the UI lists all valid tokens, whether created here or programmatically, by the Management API. As soon as a token expires, it no longer appears in the listing and count of tokens.

If any tokens exist for the particular tenant or domain, they are listed on the Tokens tab with a counter and a Filter Tokens field, which lets you search for tokens that match the string within the Owner name or Description text. For security reasons, the S3 Secret Key is never displayed in the UI after creation.

Double-click a token to view its properties and, optionally, delete it:



login

Owner	Description
admin1	Content Portal login token
admin1	Content Portal login token
admin1	Content Portal login token
admin1	Content Portal login token
admin1	Content Portal login token
admin1	Content Portal login token
admin1	Content Portal login token
admin1	Content Portal login token

dc5a5fabf058508ce18008c8dda131be

Metadata

Owner	admin1
Creation	2018-09-11
Expiration	2018-09-12
S3 Key	No
Description	Content Portal login token

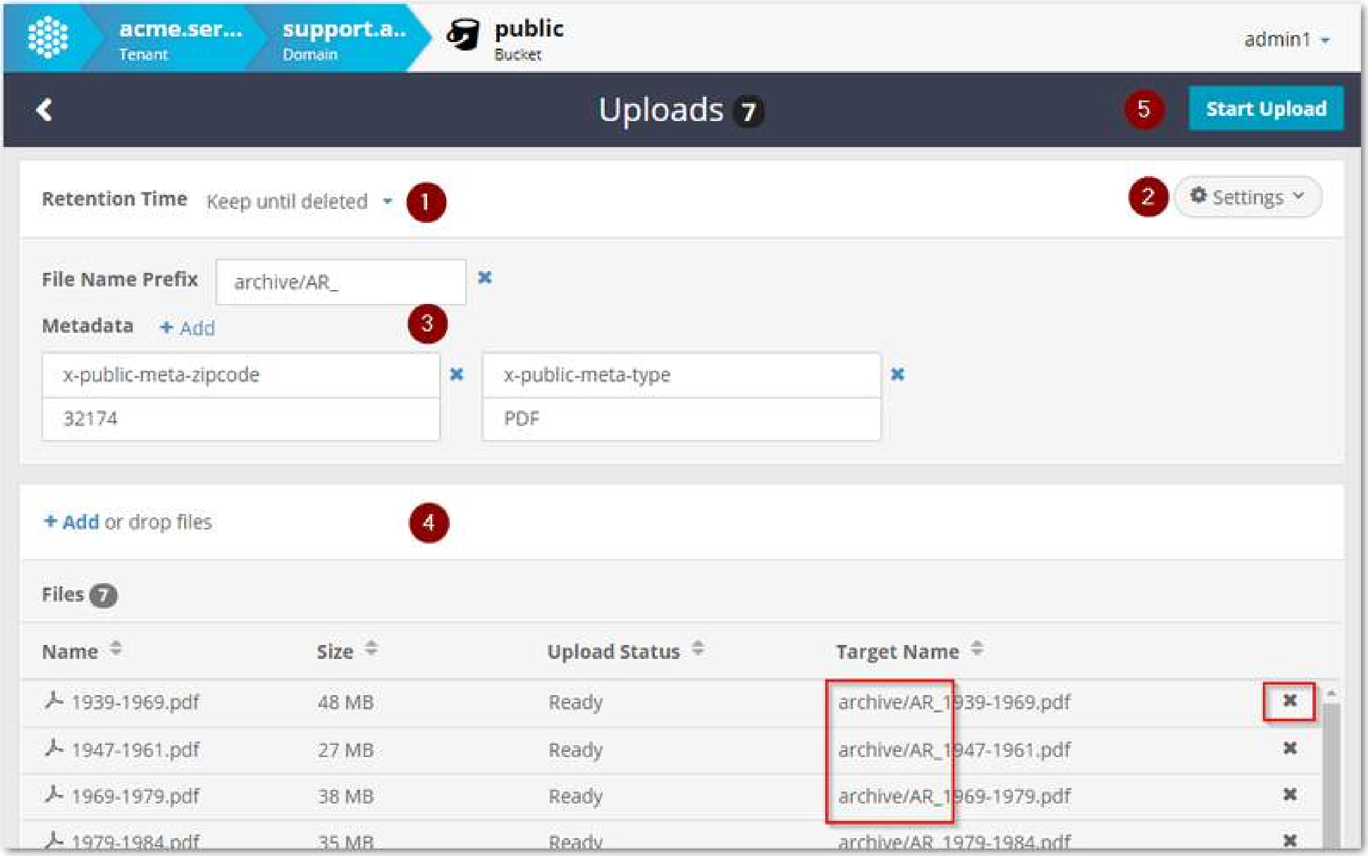
> Caution
>
> If you delete a token through this interface, you cannot restore it.

Uploading Files

Although the bulk of each tenant's content is likely to be uploaded by applications that integrate with Swarm, you can view and upload files directly from the Content UI.

> Tip
>
> The Content IDs bucket is a system-generated container that manages all unnamed objects, which are identified only by their UUID. If you upload files to this bucket, they are stored with new UUIDs with the source file name saved as the Content-Disposition metadata value.

When you upload files, you are only required to set a Retention Time, but you can expand the Settings panel, which gives you additional control over the upload. Complete the options in this order:



Retention Time: Keep until deleted

File Name Prefix: archive/AR_

Metadata:

x-public-meta-zipcode	x-public-meta-type
32174	PDF

Name	Size	Upload Status	Target Name
1939-1969.pdf	48 MB	Ready	archive/AR_1939-1969.pdf
1947-1961.pdf	27 MB	Ready	archive/AR_1947-1961.pdf
1969-1979.pdf	38 MB	Ready	archive/AR_1969-1979.pdf
1979-1984.pdf	35 MB	Ready	archive/AR_1979-1984.pdf

Tip

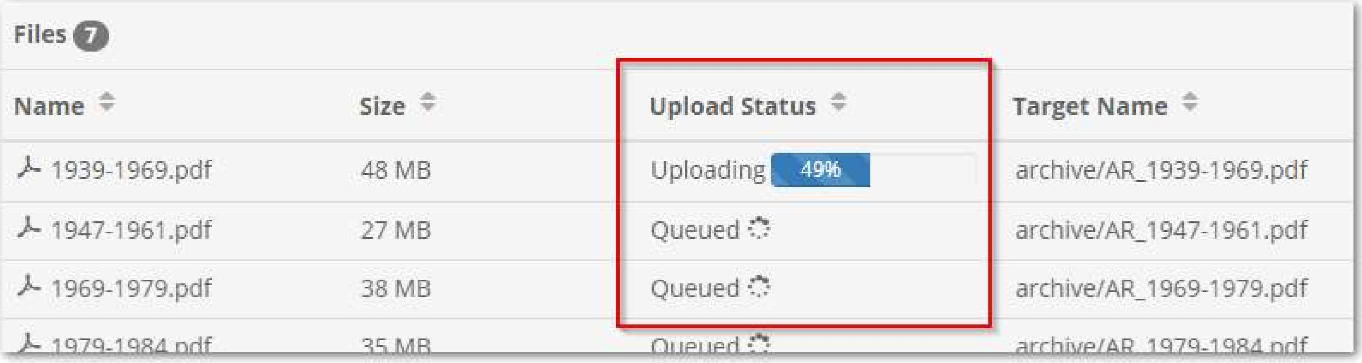
Set your upload options carefully and verify the target names before selecting Start Upload. It's faster to delete files from the upload queue than to clear out objects that were created erroneously.

Setting	Values	Example	Notes
Retention time	• Keep until deleted • 1 year • 3 years • 5 years • Date selector	3 years	Required. Defaults to Keep until deleted, which retains the object indefinitely. You may select a preset duration or pick a date after which the files will be automatically deleted by Swarm. Note If you keep the default and set no expiration, the uploaded content will not have a Lifepoint defined in its metadata.

File Name Prefix	{string}	`2016/Q3/West/`	(optional) Enter the prefix to add to the name of each file being uploaded. You can use slashes in the prefix to help organize the content, but understand that doing so does not create buckets or folders within buckets: the prefix becomes part of the content's name in Swarm. No trailing '/' is appended to the prefix, so a prefix of "2016/Q3/West" and a file name of "filename.txt" will result in an object name of "`2016/Q3/Westfilename.txt`". Note The name prefix option is not available when uploading to the Content IDs bucket because the files will be assigned a UUID.

Metadata	{custom·header} = {custom·value}	`x-status-meta = active`
		`x-public-meta = true`
			(optional) Add one or more tags to the uploaded content. Metadata is stored with every object and may be used for Collections and searching within Swarm. Metadata names must match one of these patterns:
			● `x-*-meta`
			● `x-*-meta-*`
			Tip
			See Managing Collections for how to make use of your custom metadata in searching and collections.

After you start the upload, the status of each file upload is displayed dynamically:



Files 7

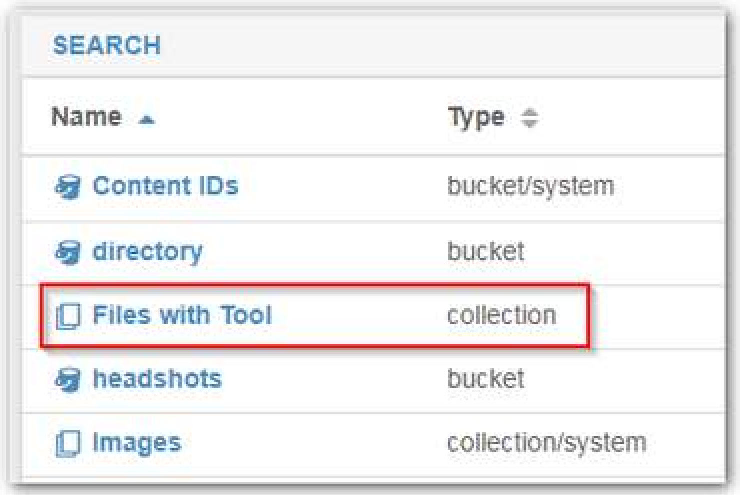
Name	Size	Upload Status	Target Name
1939-1969.pdf	48 MB	Uploading 49%	archive/AR_1939-1969.pdf
1947-1961.pdf	27 MB	Queued	archive/AR_1947-1961.pdf
1969-1979.pdf	38 MB	Queued	archive/AR_1969-1979.pdf
1979-1984.pdf	35 MB	Queued	archive/AR_1979-1984.pdf

Tip
You can cancel an individual file upload in progress by clicking the x icon at the end of the line.

Managing Collections

Collection Essentials

A collection is the result set of a search that you ran against a domain (or bucket) and then saved. You will find collections listed among the buckets at the domain level:



SEARCH

Name ▲	Type ⇕
Content IDs	bucket/system
directory	bucket
Files with Tool	collection
headshots	bucket
Images	collection/system

Collections let you shape the view of your data in three ways:

1. Scope — Set the scope of the search (either an entire domain or just a bucket)
2. Filters — Add search criteria for filtering (by name, owner, size, type, date, and/or metadata)
3. Display — Add/remove columns to display (such as to add metadata or custom metadata fields to the view)

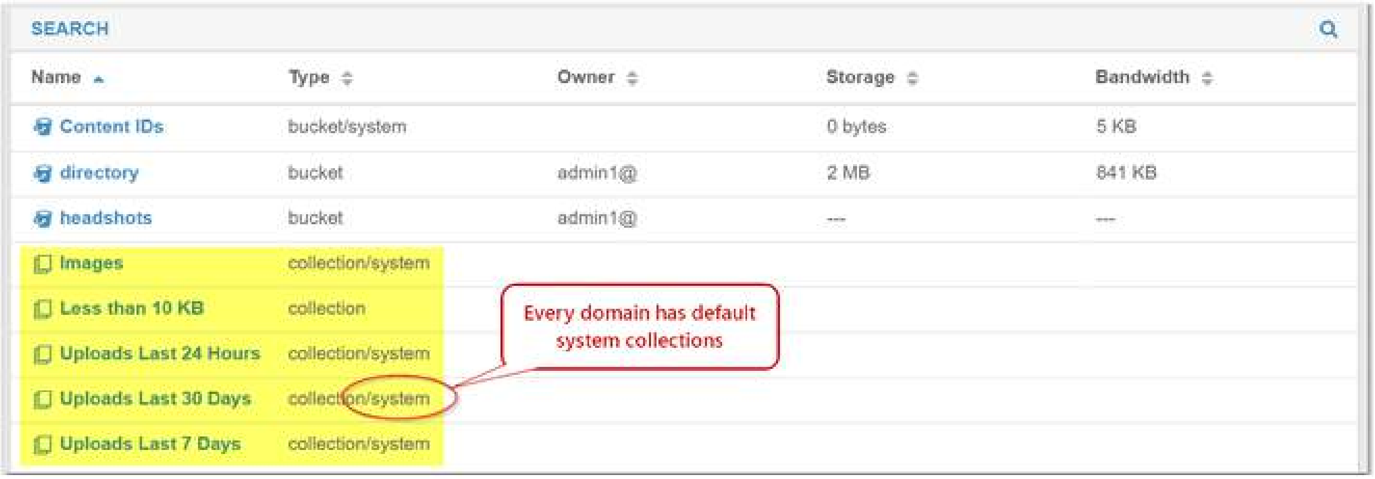
System-Created Collections

The Contents view that appears when you open a domain gives you quick access to the buckets, files, and upload activity of your domain. By default, domains include five permanent system collections for common domain-wide inquiries:

- Images — domain-wide listing of all files of Type `image`, across all buckets
- Uploads Last 24 Hours — all files uploaded in the last 24 hours, across all buckets
- Uploads Last 30 Days — all files uploaded in the last 30 days, across all buckets

- Uploads Last 7 Days — all files uploaded in the last 7 days, across all buckets

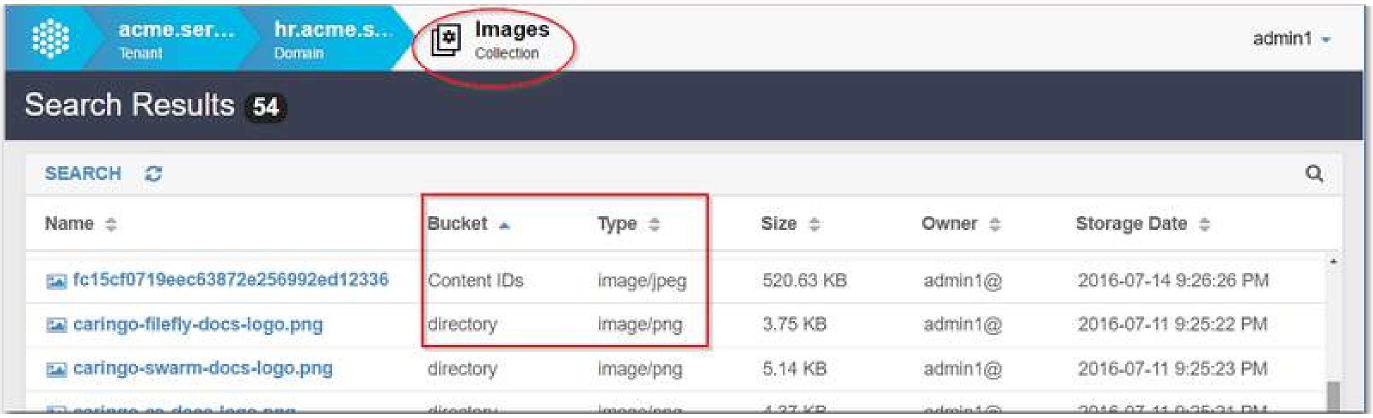
Tip
You can identify the default collections by their Type, collection/system.



SEARCH

Name	Type	Owner	Storage	Bandwidth
Content IDs	bucket/system		0 bytes	5 KB
directory	bucket	admin1@	2 MB	841 KB
headshots	bucket	admin1@	---	---
Images	collection/system			
Less than 10 KB	collection			
Uploads Last 24 Hours	collection/system			
Uploads Last 30 Days	collection/system			
Uploads Last 7 Days	collection/system			

For example, the Images collection lists all uploaded graphic files of Type "image", both named and unnamed (UUID), across all named buckets as well as the Content IDs bucket:



Search Results 54

SEARCH

Name	Bucket	Type	Size	Owner	Storage Date
fc15cf0719eec63872e256992ed12336	Content IDs	image/jpeg	520.63 KB	admin1@	2016-07-14 9:26:26 PM
caringo-filefly-docs-logo.png	directory	image/png	3.75 KB	admin1@	2016-07-11 9:25:22 PM
caringo-swarm-docs-logo.png	directory	image/png	5.14 KB	admin1@	2016-07-11 9:25:23 PM

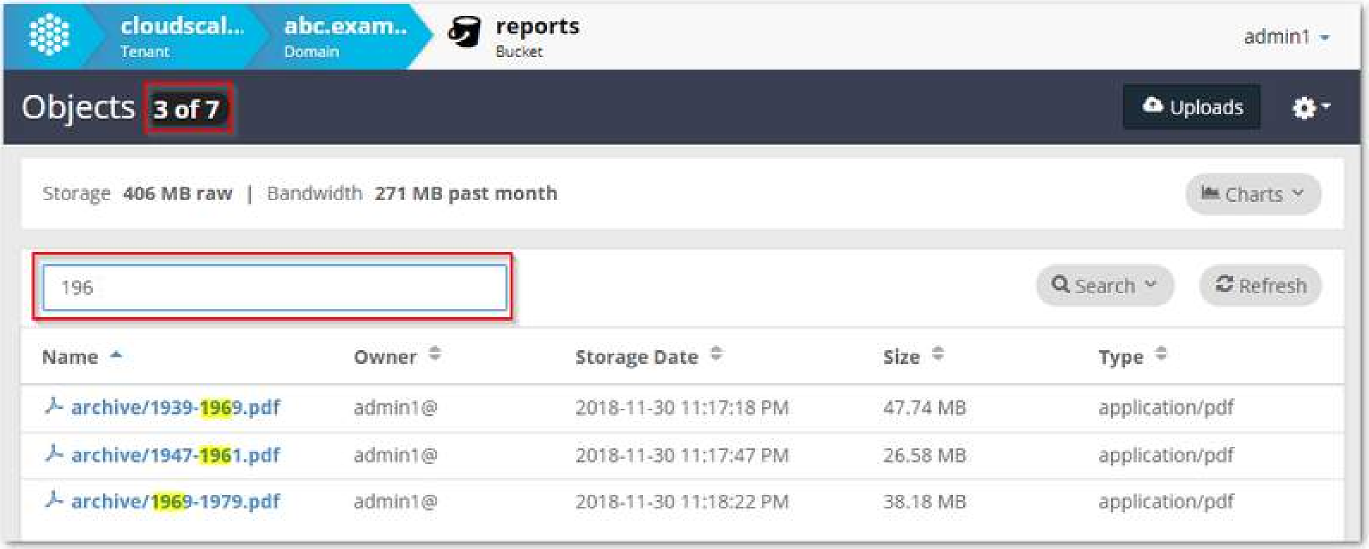
Tip
Each of the columns lets you sort, ascending and descending. Click the column header to toggle the sort direction.

To find your target faster, you can narrow the listing by entering a string in the box that filters by Name.

Although these system-generated collections cannot be changed or deleted, you can build on this set with new collections (saved searches) of your own design. If you open a permanent collection and modify it, you can save it under a new name using the Save As button.

New Search by Name

When you have a large number of objects and just want to narrow it down by name (as you do in file system searches), enter a string (no wildcards) in the Filter box. The case-insensitive search begins as soon as you begin typing: if you enter "b", any object that has the letter B (regardless of case) somewhere in its name will appear in the list.

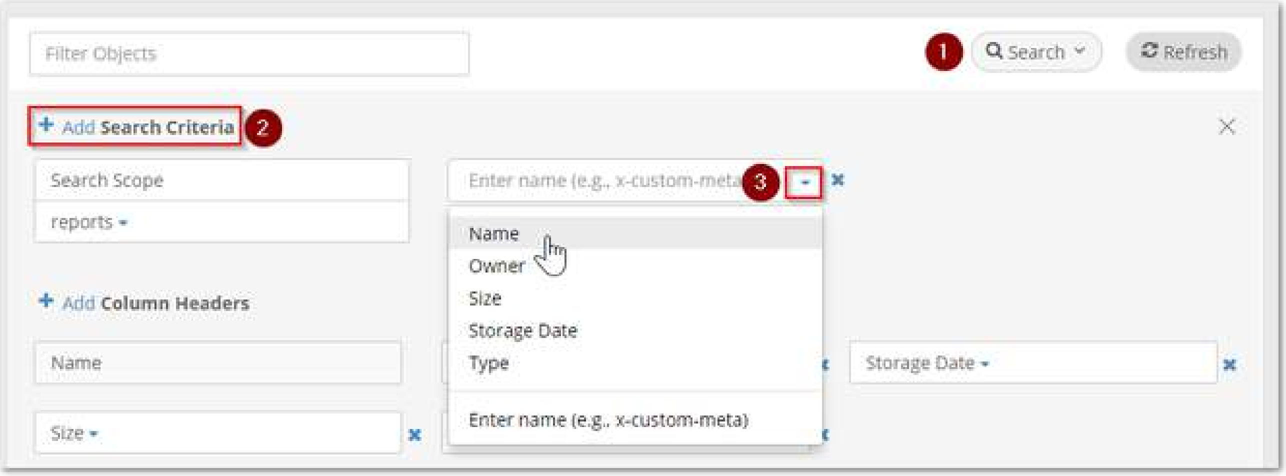


196

Name	Owner	Storage Date	Size	Type
archive/1939-1969.pdf	admin1@	2018-11-30 11:17:18 PM	47.74 MB	application/pdf
archive/1947-1961.pdf	admin1@	2018-11-30 11:17:47 PM	26.58 MB	application/pdf
archive/1969-1979.pdf	admin1@	2018-11-30 11:18:22 PM	38.18 MB	application/pdf

The Objects count will show what portion of the total listing matches your string (here, 3 of 7).
If this is a search that you will want to repeat or make available for others, you'll need the full Search controls:

1. Delete the string in the Filter Objects box and click the Search button to open the search pane.
2. Click + Add Search Criteria, which adds a new (empty) criteria operation.
3. From the drop-down list, select Name.



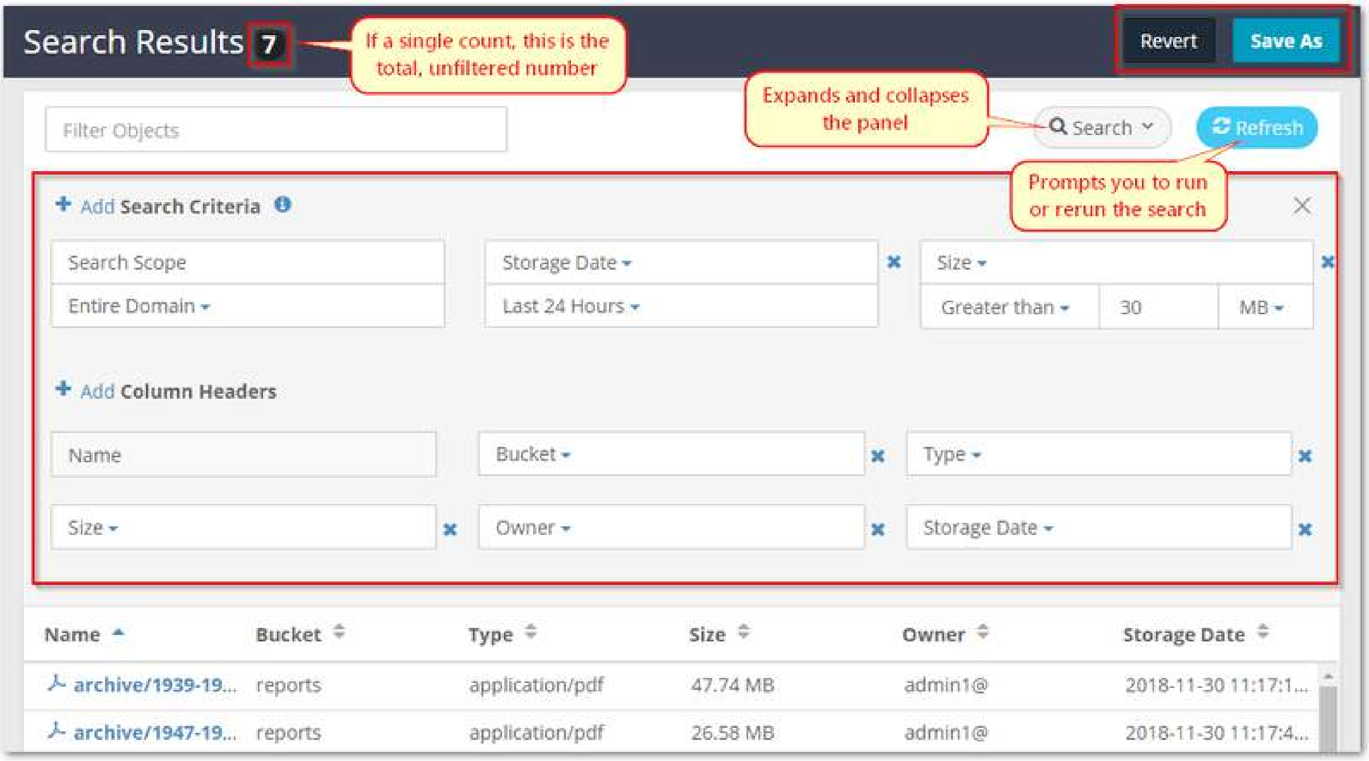
4. When prompted for the Value to match on Name, enter the string that you had used in the Filter Objects box.
5. The Refresh button will flash to prompt you to run the search. Click Refresh to verify that the search returns the same objects as before.
6. At top, click Save As, and name your new collection.

Using the Search Panel

The Collections feature lets you perform complex ad hoc searches and define custom saved searches and views.

> Context dependence
> Keep in mind that your collections depend on the existence of containers (bucket, domain). If the bucket was renamed or recreated, edit your collection to update it. If the original container was deleted, update or delete the collection.

In the listing for every collection, the Search Results panel appears with a results counter:



If a single count, this is the total, unfiltered number

Expands and collapses the panel

Prompts you to run or rerun the search

Search Scope: Entire Domain

Storage Date: Last 24 Hours

Size: Greater than 30 MB

Column headers: Name, Bucket, Type, Size, Owner, Storage Date

Name	Bucket	Type	Size	Owner	Storage Date
archive/1939-19...	reports	application/pdf	47.74 MB	admin1@	2018-11-30 11:17:1...
archive/1947-19...	reports	application/pdf	26.58 MB	admin1@	2018-11-30 11:17:4...

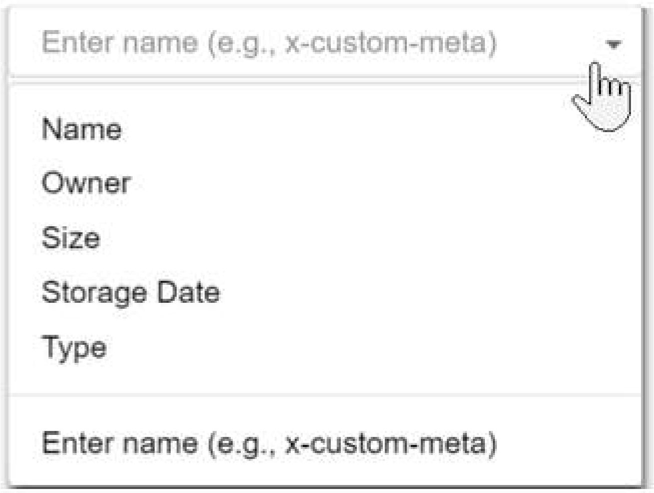
The search commands have these effects:

Search	Toggles the search panel (which defines what to search on and what columns to return) in and out of view, above the search results. Collapsing the panel from view does not change the settings.
Refresh	Runs or reruns the current search definition. The flashing is a prompt for you to rerun the search because changes are detected.
Filter	Performs string matching on the names of objects, including the GUIDs of unnamed objects.
Delete	Appears only for custom saved collections. System collections cannot be deleted.
Revert	Discards your current changes to an existing definition. Use this for ad hoc searches, to avoid keeping unneeded collections.

Save As	Saves your current definition (scope, criteria, and columns) for later use. Note When creating a collection, follow the Naming Rules for Swarm for named objects and make sure that it is unique to the domain.

Setting Search Criteria

You can define search criteria against basic metadata, extended metadata, and your own custom metadata. Click the + Add button as many times as needed to combine search criteria that narrow the results to the data you want to see.


Enter name (e.g., x-custom-meta)
Name
Owner
Size
Storage Date
Type
Enter name (e.g., x-custom-meta)

Search on Common Metadata

Several commonly searched attributes are predefined for ready access:

Name	Units/Range	Notes
Name	● None	Use wildcards to specify the string to match on: `*cert*`
Owner	● None	Use wildcards to specify the string to match on: `*admin*`

Size	- bytes - KB - kilobytes - MB - megabytes - GB - gigabytes - TB - terabytes - PB - petabytes - EB - exabytes	Select the operation for the comparison: - Greater than - Equals - Less than
Storage Date	- Last 24 Hours - Last 7 Days - Last 30 Days - Last (custom) - Before... - Since...	
Type	- Audio - Image - PDF - Text - Video - None - Enter value (such as `image/jpeg`)	

Searching Extended and Custom Metadata

You can also search against any system or custom metadata that is stored with each object. The following shows common metadata included on an object's detail view:

Metadata	Example value	Notes
Size	117.12 KB	
Type	image/jpeg	
Owner	admin1@	
Stored Date	2015-09-23 5:57:25 PM	
Castor-System-Cid	7da76343ad6bc9f2f739f0595a2756e4	
Castor-System-Cluster	raindance	
Castor-System-Created	2015-09-23 5:57:25 PM	

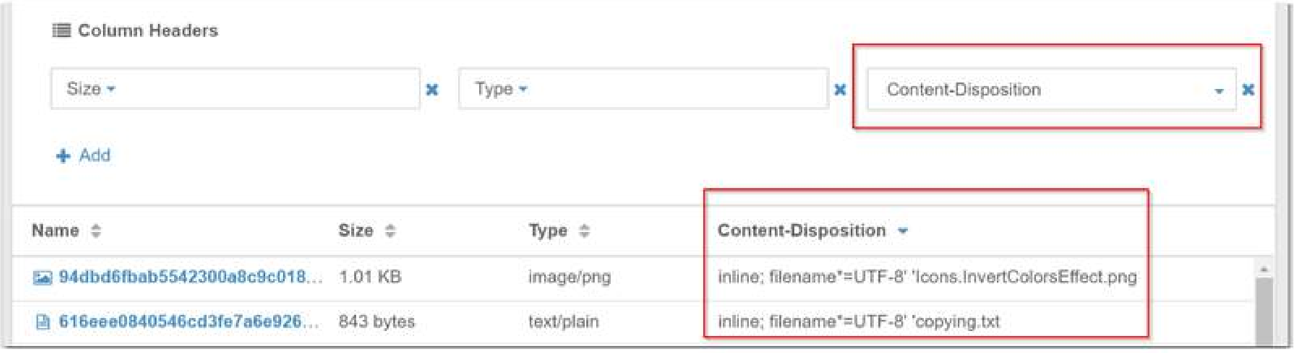
Castor-System-Name	jsmith.jpg	
Castor-System-Version	1443049045.780	
Content-Disposition	attachment; filename="jsmith.jpg"	Stores the original name of the source file that was uploaded.
Content-Md5	5QET59jX1t8//iD4CgnWWQ==	
Etag	"9dbfd0d4b524e8914280b0b1f7d12e3b"	
Lifepoint	[Tue, 29 Sep 2015 05:00:00 GMT] deletable, [] delete	Stores the lifepoint settings in force for the object, if any exist. The example shows that the object was imported with a specific expiration date.
X-Last-Modified-By-Meta	admin1@	
X-<custom-tag-name>-Meta	2008-01-15 12:00:00 AM	Custom metadata tags that were entered when the file was stored. The example shows hire date data that corresponds to this custom tag: `X-Hiredate-Meta`

Example Metadata Search

> Tip
> To search for results that have the metadata tag but with no associated value, specify a value that is only white space. Leading or trailing white space in text strings for names or metadata tags are ignored.

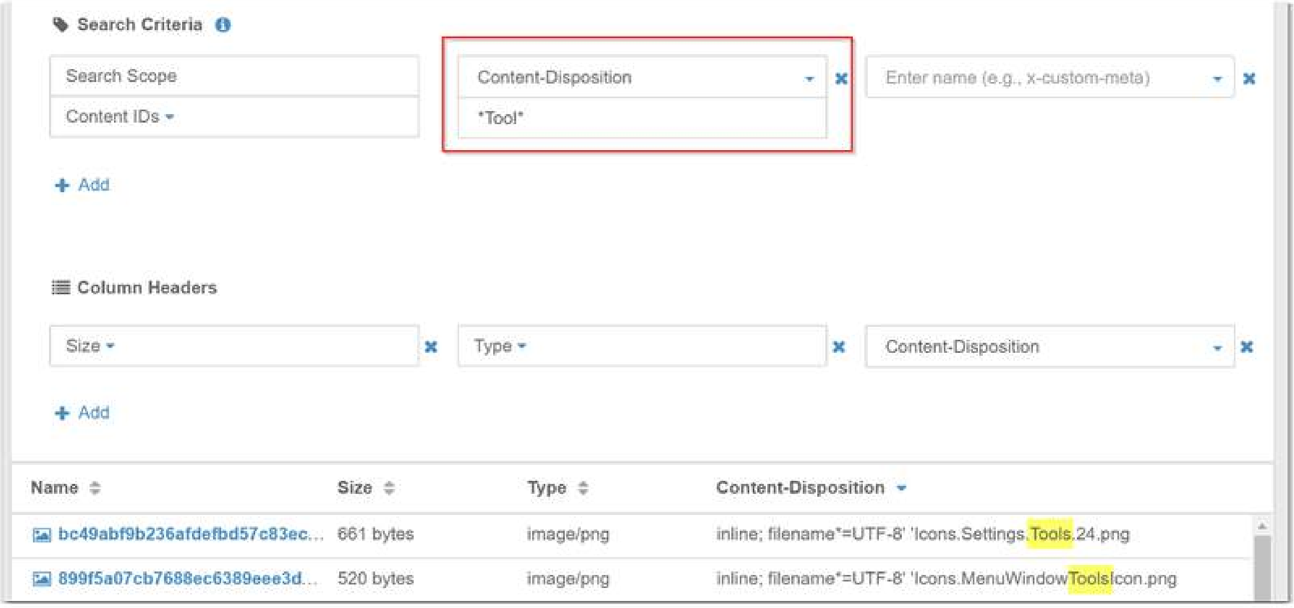
Suppose a set of files were uploaded into the Content IDs bucket, so that they were stored by UUID. In that case, the original filenames are stored as metadata. If you wanted to be able to see what the source filename was for each image, you would take these steps:

1. From the domain list, open the Content IDs bucket.
2. In the Column Headers section, click +Add.
3. Type in the name of the metadata field that stores the source filename: Content-Disposition.
4. Click Refresh to populate the new column:



Name	Size	Type	Content-Disposition
94dbd6fbab5542300a8c9c018…	1.01 KB	image/png	inline; filename*=UTF-8' 'Icons.InvertColorsEffect.png
616eee0840546cd3fe7a6e926…	843 bytes	text/plain	inline; filename*=UTF-8' 'copying.txt

5. To narrow the results to only the files that had "Tool" in the name originally, create a Search Criteria on Content-Disposition, then Refresh:



Content-Disposition: *Tool*

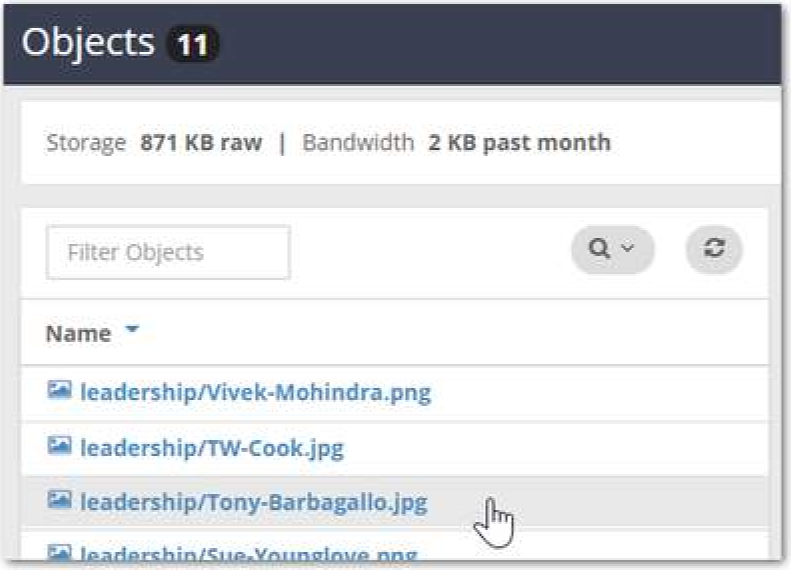
Name	Size	Type	Content-Disposition
bc49abf9b236afdefbd57c83ec…	661 bytes	image/png	inline; filename*=UTF-8' 'Icons.Settings.Tools.24.png
899f5a07cb7688ec6389eee3d…	520 bytes	image/png	inline; filename*=UTF-8' 'Icons.MenuWindowToolsIcon.png

6. To keep this collection, select Save As and provide a name for the collection for future reference; otherwise, click Revert.

Searching by Metadata Selected from Objects

There is tremendous utility in building search collections based on metadata, especially the extended metadata that Swarm indexes and your own custom metadata. For searching this kind of metadata, the easiest method is to start from an object that has the metadata you want.

1. Find and double-click the object to view its details:



Objects 11

Storage 871 KB raw | Bandwidth 2 KB past month

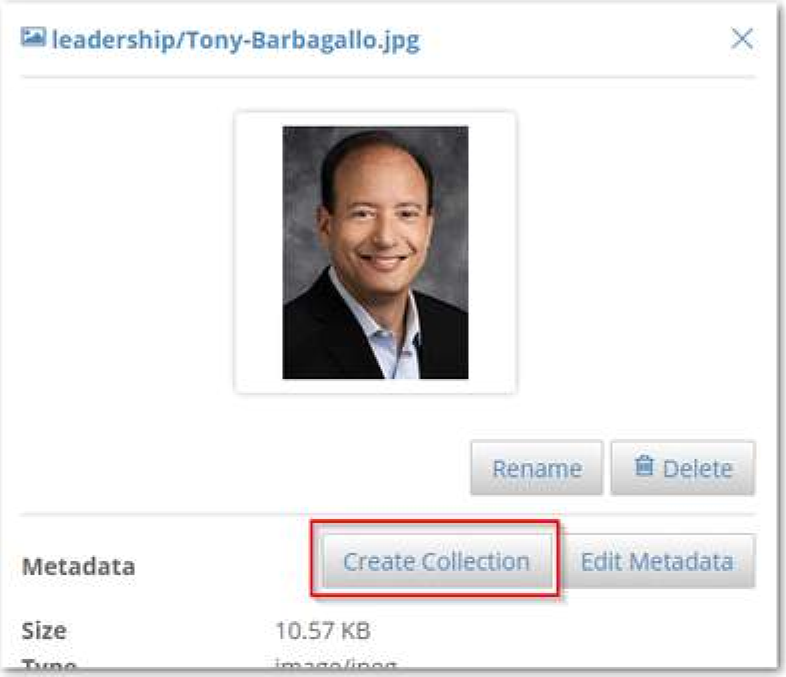
leadership/Vivek-Mohindra.png

leadership/TW-Cook.jpg

leadership/Tony-Barbagallo.jpg

leadership/Sue-Younglove.png

2. In the detail view, click Create Collection:



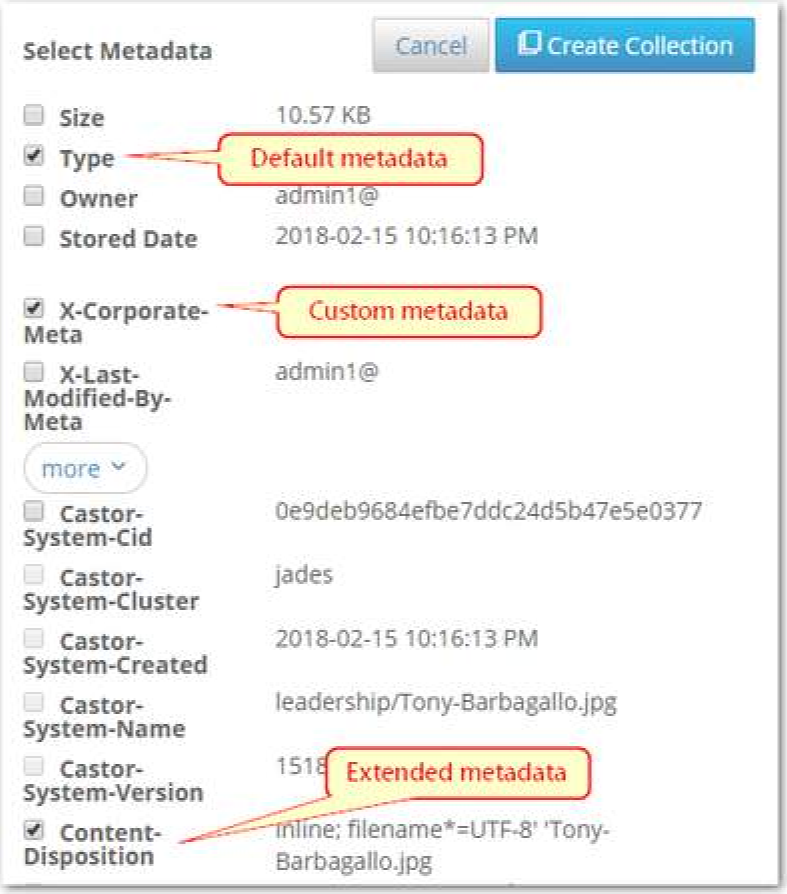
leadership/Tony-Barbagallo.jpg

Metadata

Size 10.57 KB

Type image/jpeg

3. Select which fields you want, and then select Create Collection:



Select Metadata

☐ Size 10.57 KB
☑ Type
☐ Owner admin1@
☐ Stored Date 2018-02-15 10:16:13 PM

☑ X-Corporate-Meta
☐ X-Last-Modified-By-Meta admin1@

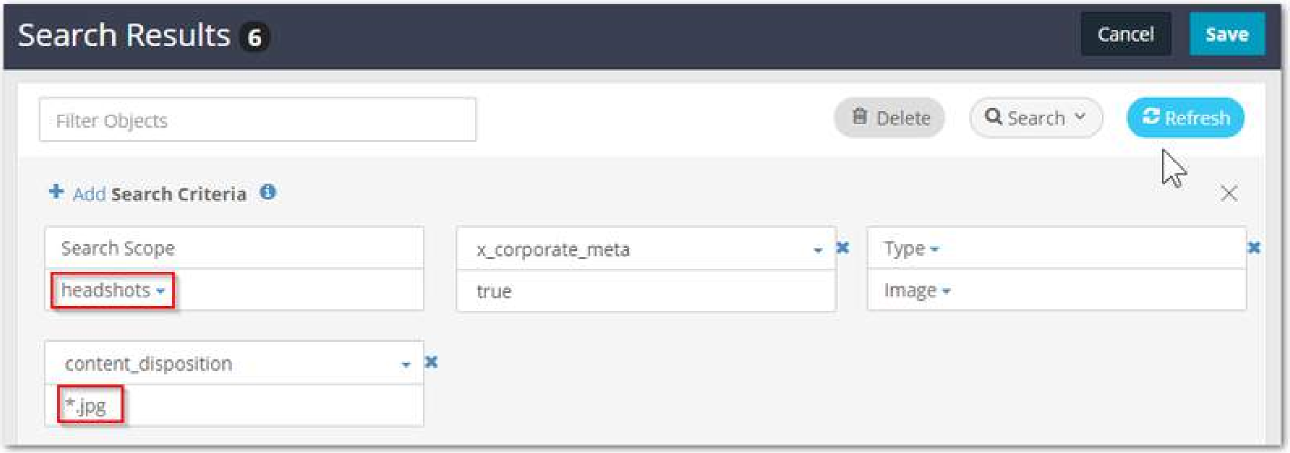
☐ Castor-System-Cid 0e9deb9684efbe7ddc24d5b47e5e0377
☐ Castor-System-Cluster jades
☐ Castor-System-Created 2018-02-15 10:16:13 PM
☐ Castor-System-Name leadership/Tony-Barbagallo.jpg
☐ Castor-System-Version 1518
☑ Content-Disposition inline; filename*=UTF-8' 'Tony-Barbagallo.jpg

Default metadata

Custom metadata

Extended metadata

4. In the Search Results, edit the Search Criteria, and click Refresh to test the results:



Search Results 6

Filter Objects

Search Scope: headshots

x_corporate_meta: true

Type: Image

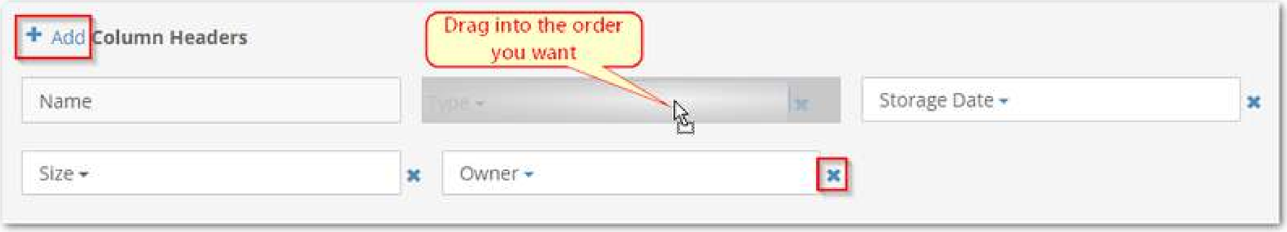
content_disposition: *.jpg

Tip

To keep the visual display of lists manageable, lists are truncated to 10,000 objects. To return a shorter list, apply more filtering.

5. When the filtering returns the correct results, you can improve the listing display by changing the Column

Headers as needed (move, add, delete).

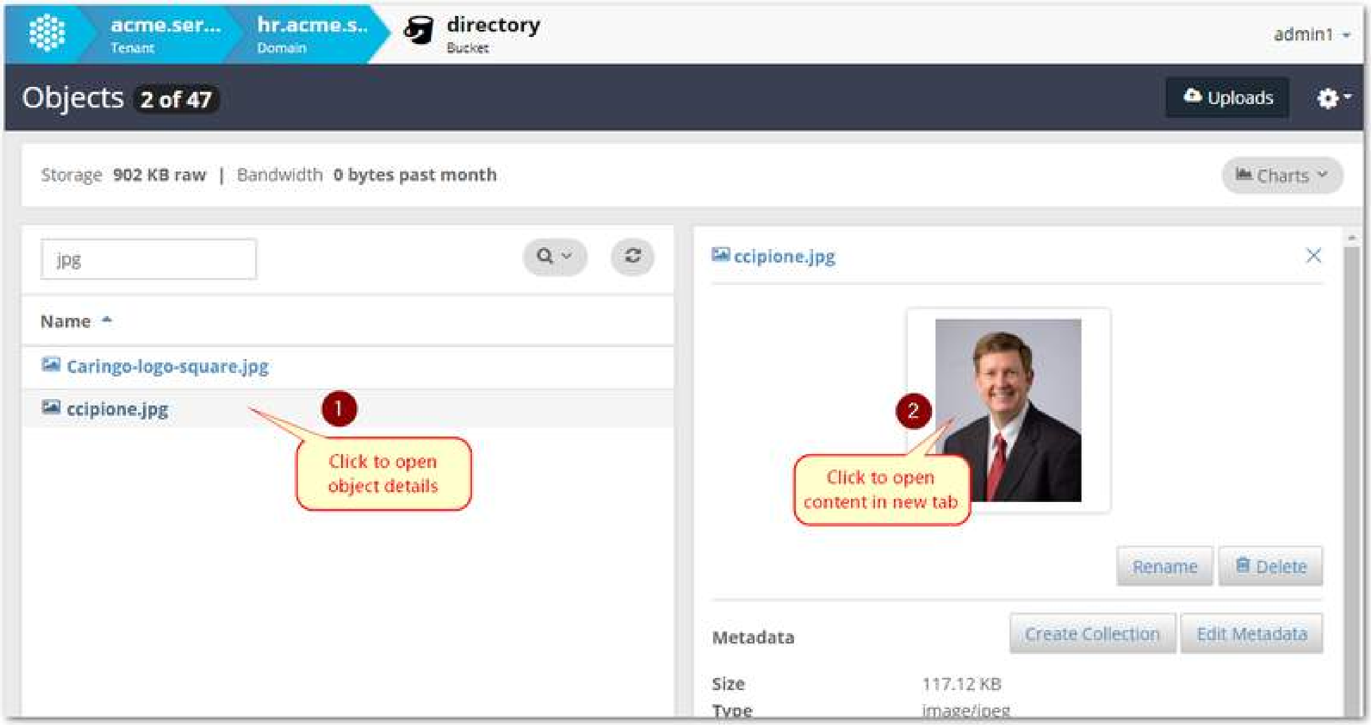


Downloading Content

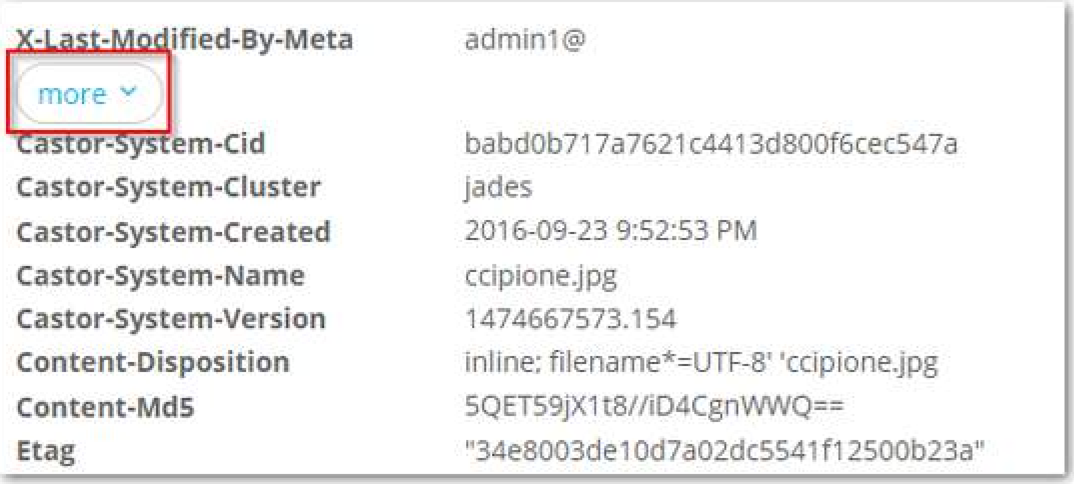
- Viewing Content
- Custom Metadata
- Deleting Content
- Downloading Content

Viewing Content

To access the details about any uploaded file, locate it in the listing and click on it. The details display at right:



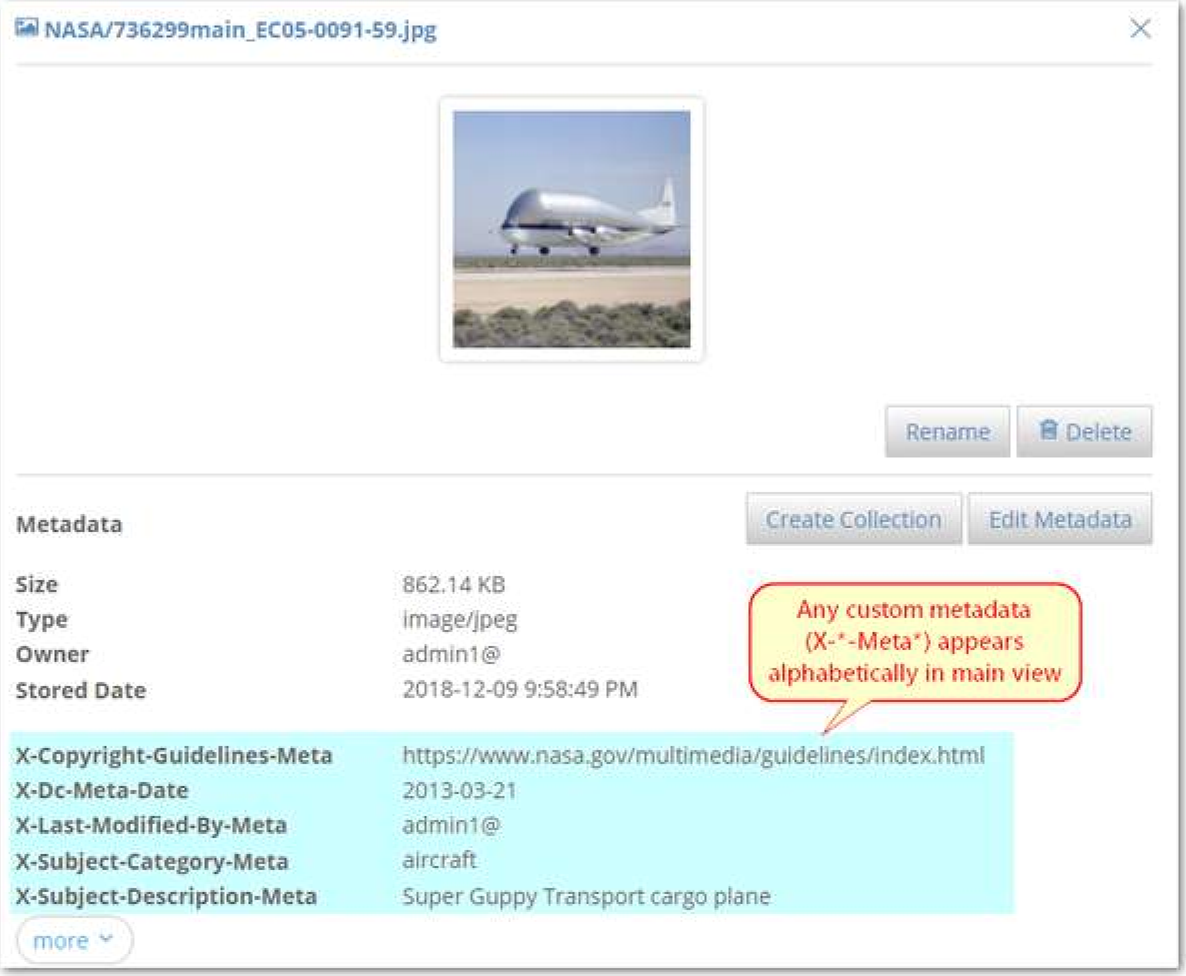
Click the more button to see what advanced or Swarm-specific metadata was stored with the file:



X-Last-Modified-By-Meta	admin1@
Castor-System-Cid	babd0b717a7621c4413d800f6cec547a
Castor-System-Cluster	jades
Castor-System-Created	2016-09-23 9:52:53 PM
Castor-System-Name	ccipione.jpg
Castor-System-Version	1474667573.154
Content-Disposition	inline; filename*=UTF-8' 'ccipione.jpg
Content-Md5	5QET59jX1t8//iD4CgnWWQ==
Etag	"34e8003de10d7a02dc5541f12500b23a"

Custom Metadata

Any custom metadata that was stored with the file appears in this main view, when you open the object details:



NASA/736299main_EC05-0091-59.jpg

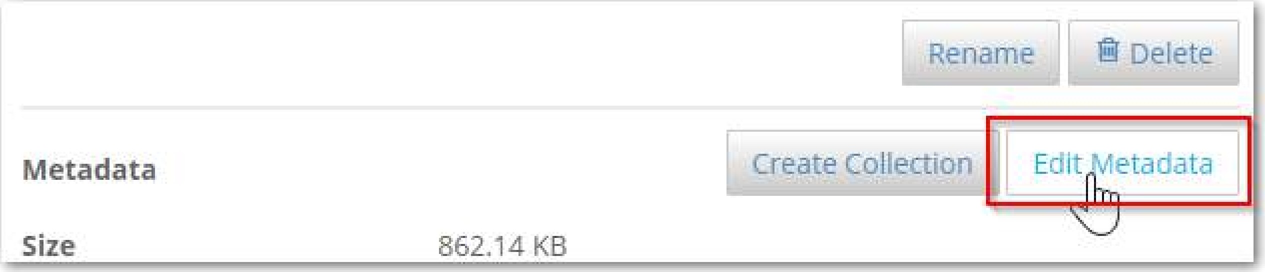
Metadata

Size	862.14 KB
Type	image/jpeg
Owner	admin1@
Stored Date	2018-12-09 9:58:49 PM
X-Copyright-Guidelines-Meta	https://www.nasa.gov/multimedia/guidelines/index.html
X-Dc-Meta-Date	2013-03-21
X-Last-Modified-By-Meta	admin1@
X-Subject-Category-Meta	aircraft
X-Subject-Description-Meta	Super Guppy Transport cargo plane

Any custom metadata (X-*-Meta*) appears alphabetically in main view

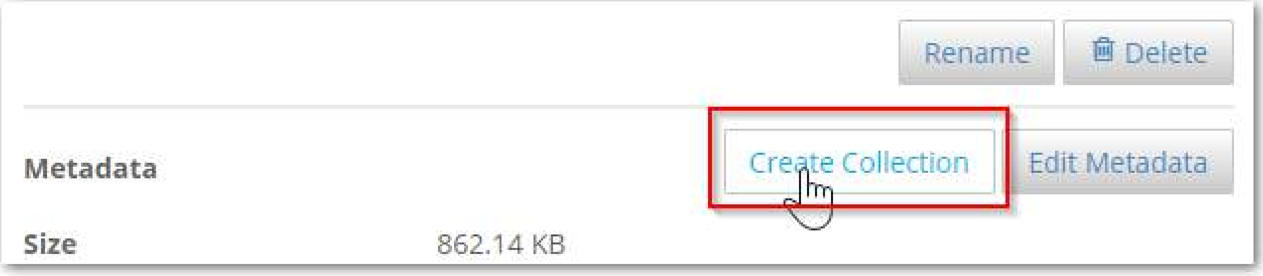
Adding custom metadata

Although custom metadata is typically added programmatically during ingest, you can correct it and even add to it using the Edit Metadata command:

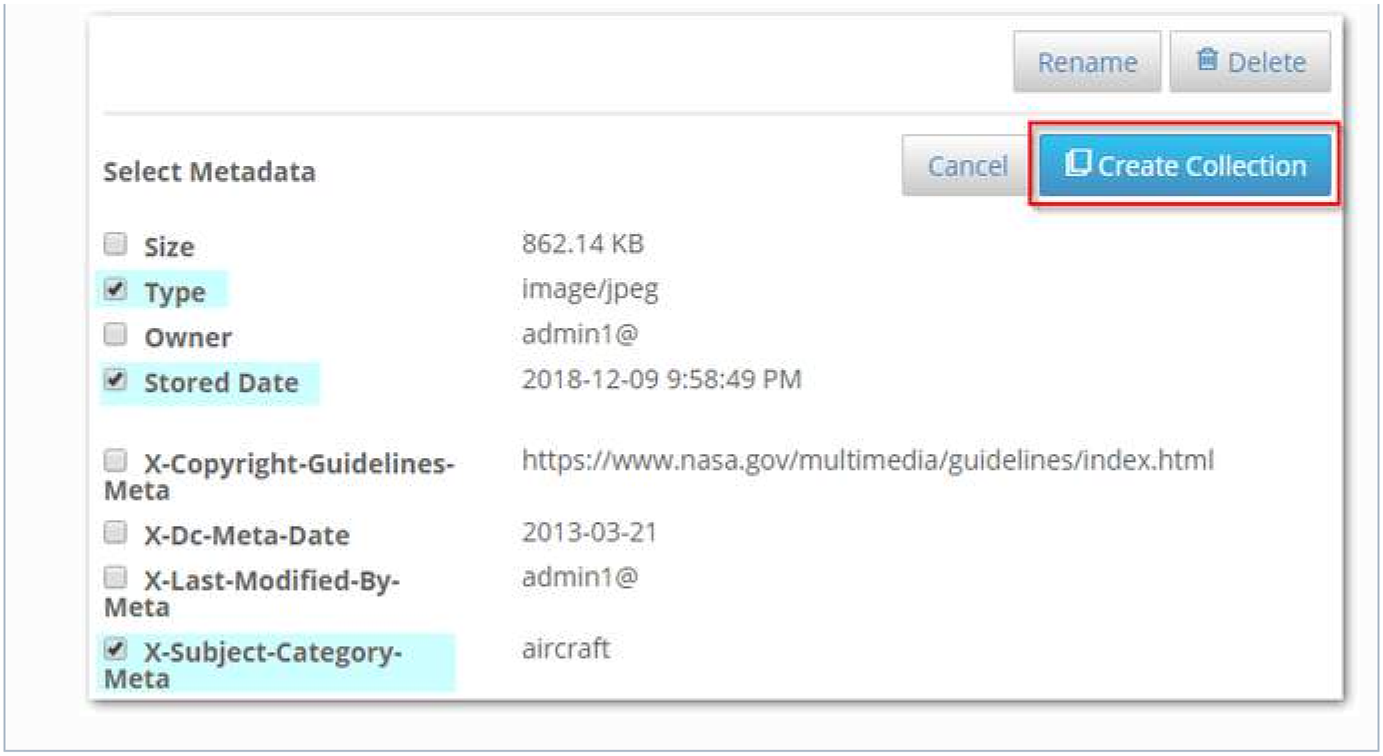


Searching custom metadata

To search on metadata, use the Create Collection command:



Check the boxes for which metadata headers you want to have available to filter in your search:

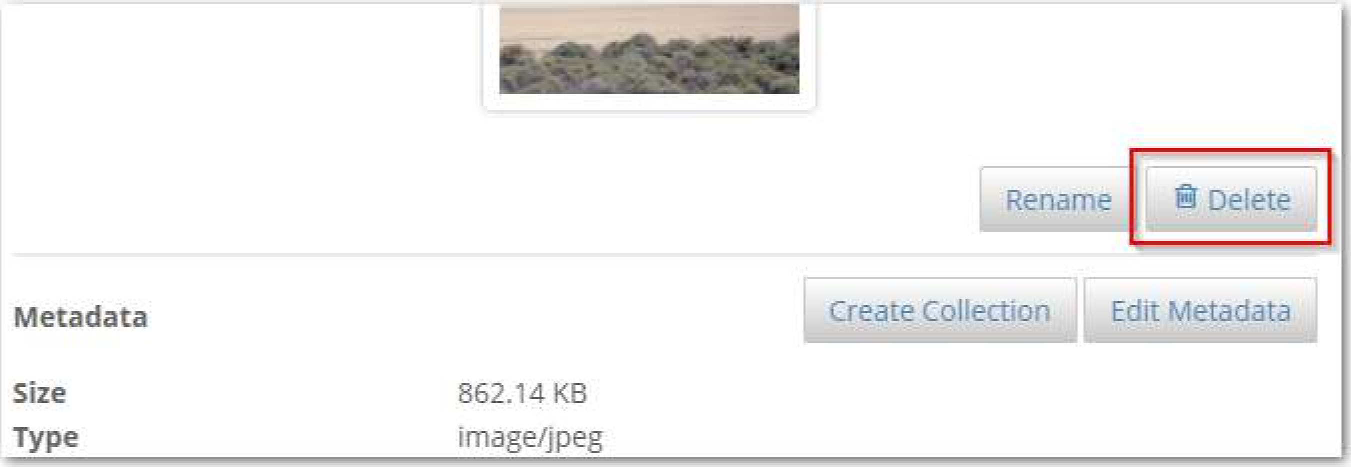


Select Metadata

□ Size	862.14 KB
☑ Type	image/jpeg
□ Owner	admin1@
☑ Stored Date	2018-12-09 9:58:49 PM
□ X-Copyright-Guidelines-Meta	https://www.nasa.gov/multimedia/guidelines/index.html
□ X-Dc-Meta-Date	2013-03-21
□ X-Last-Modified-By-Meta	admin1@
☑ X-Subject-Category-Meta	aircraft

Deleting Content

Click the Delete button to delete the object, but proceed with caution as this action cannot be undone.



Metadata

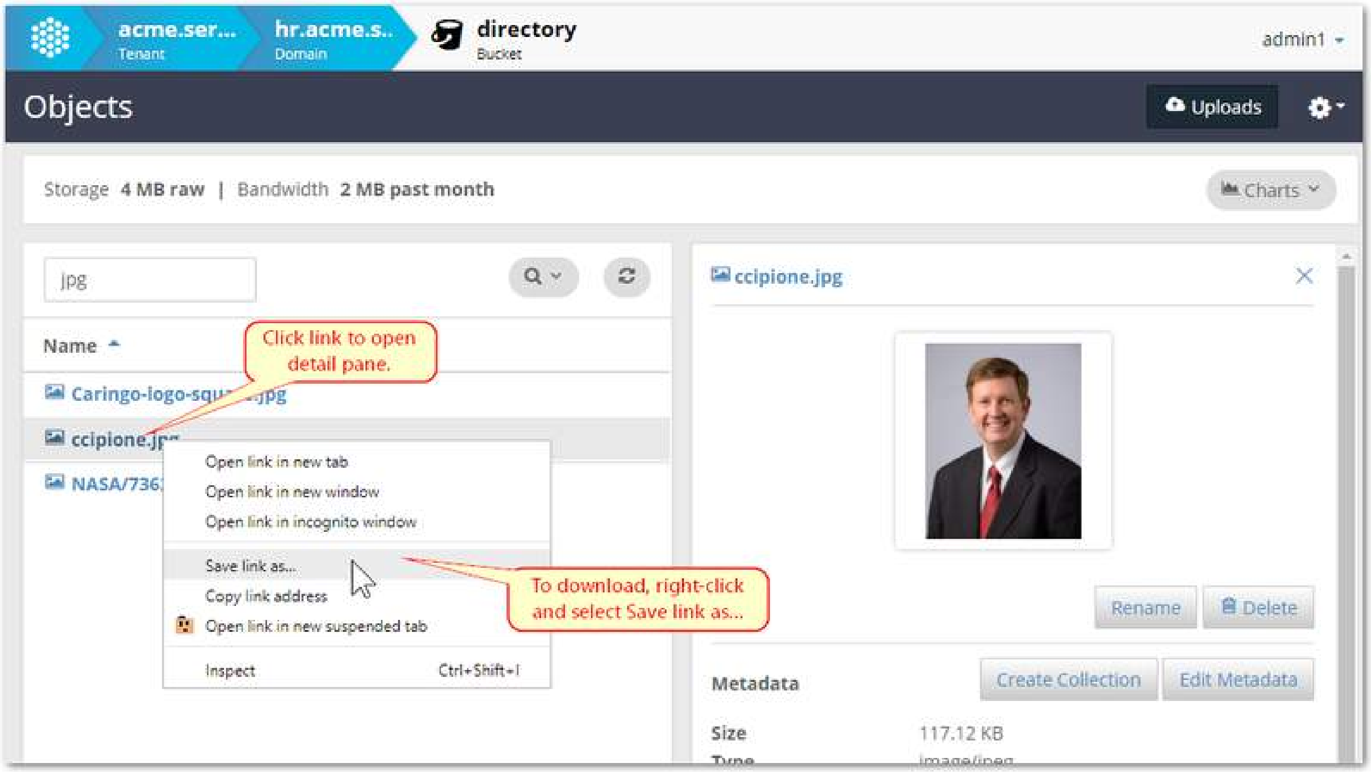
Size	862.14 KB
Type	image/jpeg

Important
Although deleted objects may continue to appear in Collection listings temporarily after they are deleted, they are no longer accessible in the storage cluster.

Downloading Content

Use your browser commands to download a local copy of an object.

To view	In the collection or bucket listing, click the object name, then click on the thumbnail, at right.
To get link	Right-click on the object name and select Copy link address or Copy link location.
To download	Right-click on the object name and select Save link as...



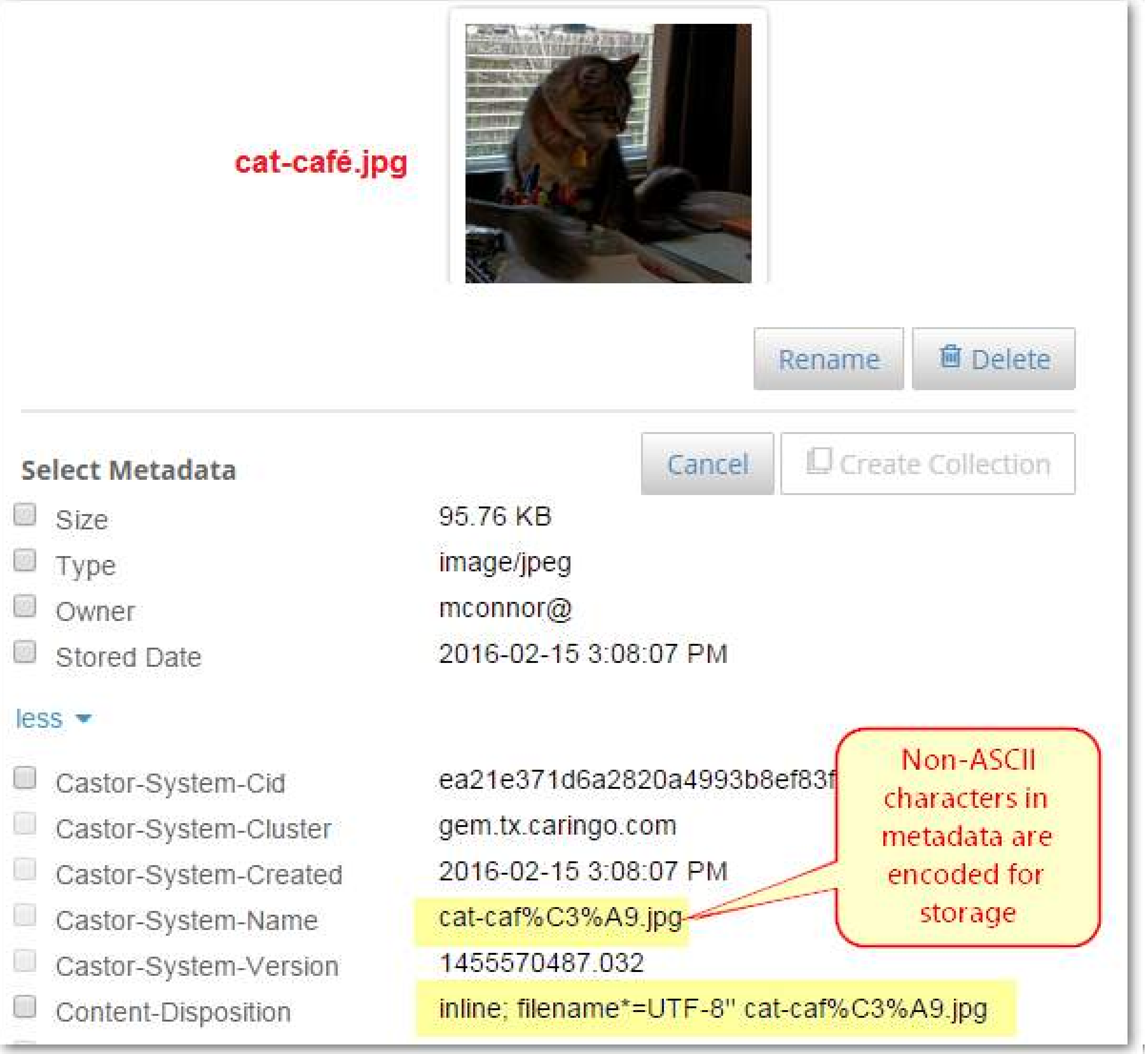
Click link to open detail pane.

To download, right-click and select Save link as...

Metadata Encoding

Any non-ASCII characters in your metadata are encoded for storage. To comply with HTTP/1.1 per RFC-2616, Swarm encodes header field content according to the rules of RFC-2047, which means that characters in sets other than ISO-8859-1 must be encoded.

What this means is that you might see encoding of non-ASCII characters in the metadata details for an object that you view in the Content UI:



cat-café.jpg

Select Metadata

□	Size	95.76 KB
□	Type	image/jpeg
□	Owner	mconnor@
□	Stored Date	2016-02-15 3:08:07 PM
□	Castor-System-Cid	ea21e371d6a2820a4993b8ef83f
□	Castor-System-Cluster	gem.tx.caringo.com
□	Castor-System-Created	2016-02-15 3:08:07 PM
□	Castor-System-Name	cat-caf%C3%A9.jpg
□	Castor-System-Version	1455570487.032
□	Content-Disposition	inline; filename*=UTF-8'' cat-caf%C3%A9.jpg

Non-ASCII characters in metadata are encoded for storage

Swarm allows all string-typed header fields to have multiple lines as well as encoded words. Swarm stores the header value as-is with the object metadata. Only when Swarm needs to use that value (such as for metadata indexing) does it decode the value. Swarm decodes header fields into Unicode and then operates on the decoded values. The original encoded persistent headers, however, remain safely stored with the object and are returned when you retrieve the object.

> For details about Swarm's handling of metadata, see Encoding Non-ASCII Characters in Metadata.

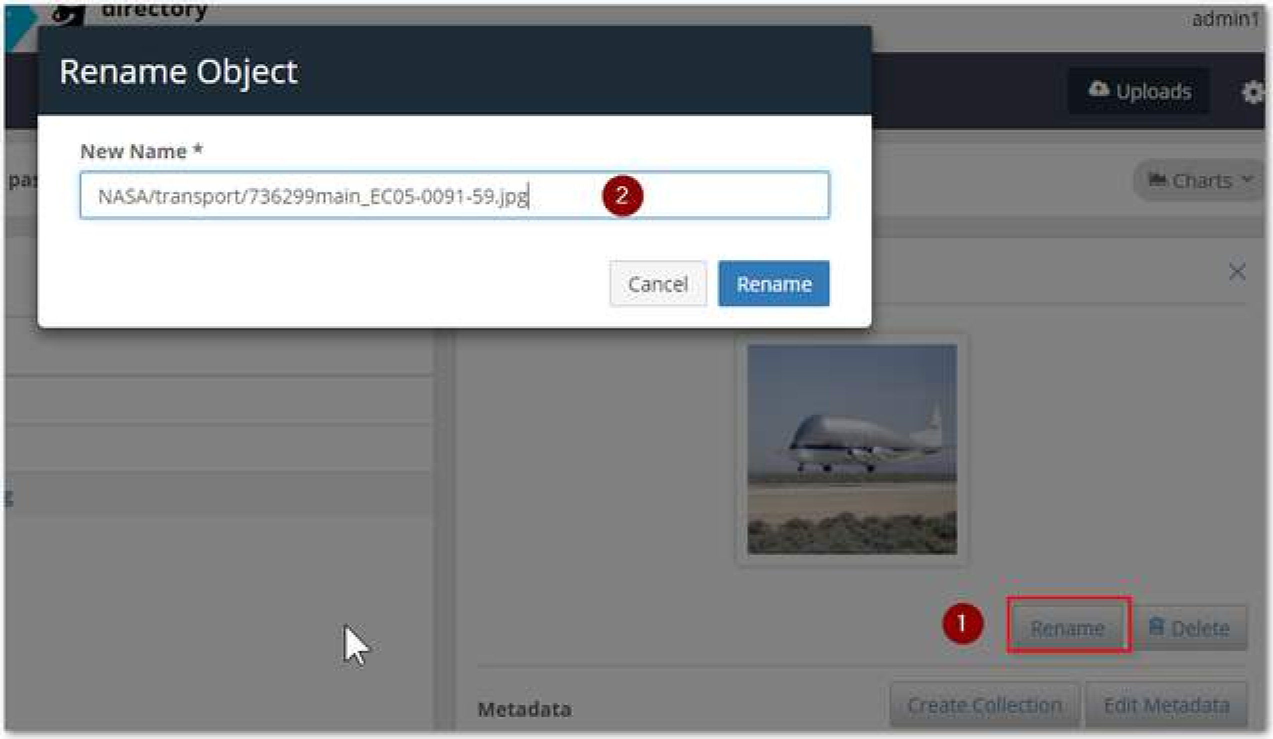
Editing Names, Metadata, and Versions

- Renaming Objects
- Editing Metadata
- Viewing and Deleting Versions

Renaming Objects

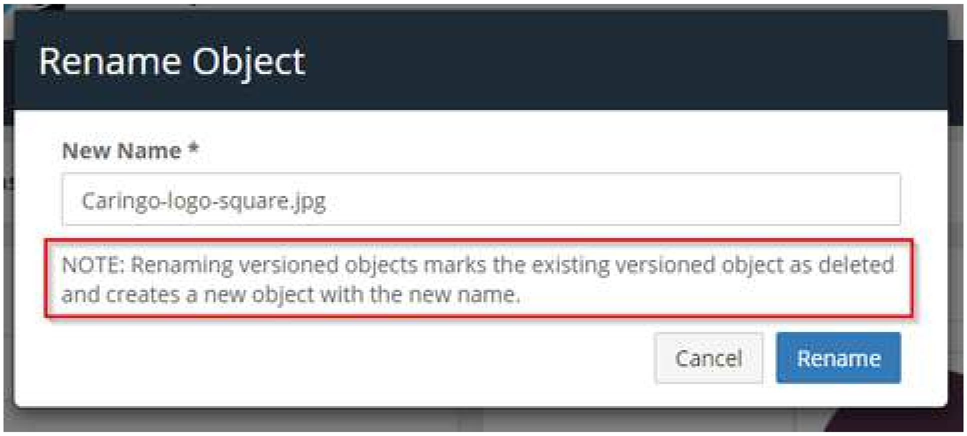
When you select and view an individual object, you can change its object name using the Rename command. (v5.5) Keep in mind that the name includes any pseudo directories that may have been added on upload to Swarm, as well as any file extensions that were retained.

> Tip
> The original filename of the uploaded object is always preserved in the metadata. Under the Metadata section, select more and locate the Castor-System-Name field.



New Name: NASA/transport/736299main_EC05-0091-59.jpg

Versioned objects — Renaming a versioned object has the effect of marking the existing object as deleted (which ends one chain of historical versions) and creating a new object using the new name (which begins a new chain of historical versions). If you are attempting to rename a versioned object, you will see a reminder of the impact of versioning:



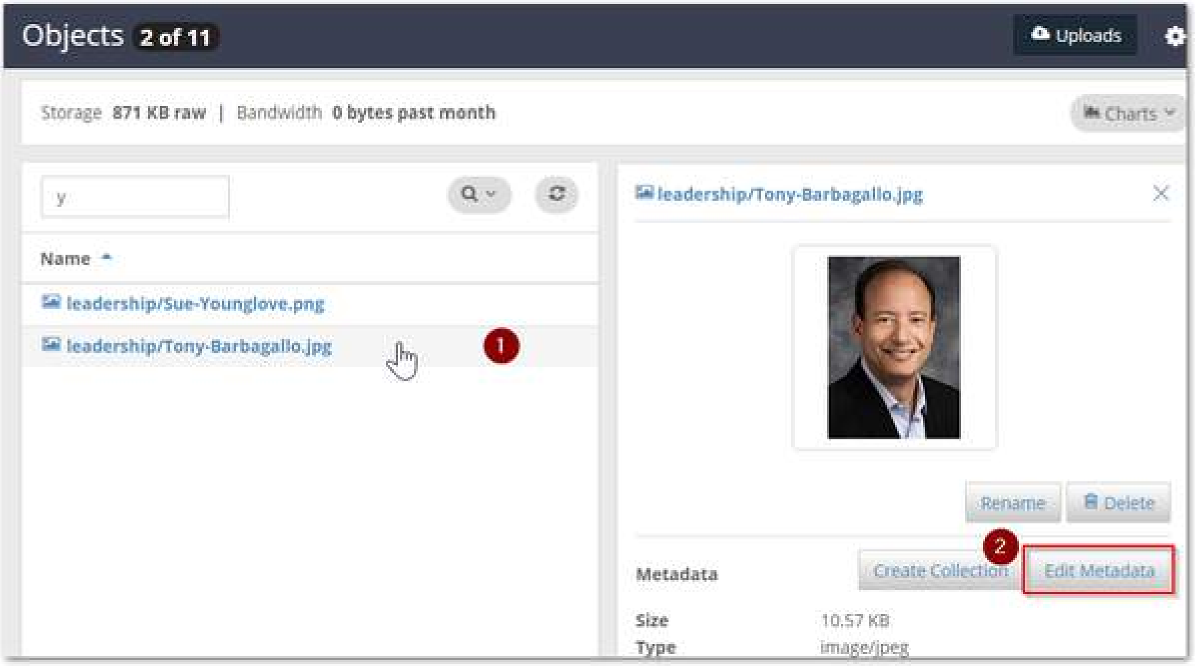
New Name: Caringo-logo-square.jpg

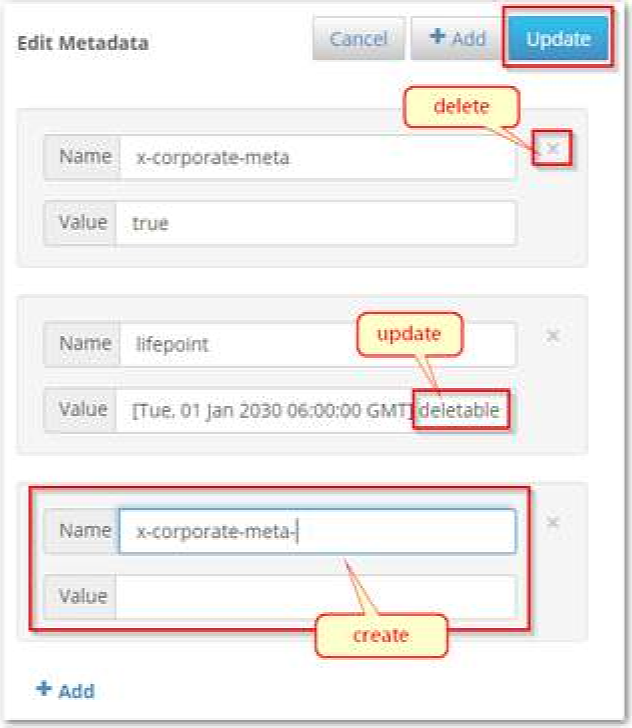
NOTE: Renaming versioned objects marks the existing versioned object as deleted and creates a new object with the new name.

Editing Metadata

The Content UI allows you to add and edit custom object metadata directly in the UI. While applications can also add and change object metadata, this interface allows you to correct and extend metadata for a given object from the convenience of your browser. If an editable metadata field (such as `x-acme-meta-color`) exists for the object, it will appear in the object's detail view, and you can change its value (`red` to `blue`) or remove it entirely. (v5.4.0)

> Versioning
>
> If versioning is enabled for the object's domain or bucket, be aware that you are only ever editing the metadata for the current version of the object. If you make several changes to metadata and save them, one new version is created. If you then add another metadata change and save it, yet another version is created, with both sets of metadata.


Objects 2 of 11
Uploads
Storage 871 KB raw | Bandwidth 0 bytes past month
Charts
y
Name
leadership/Sue-Younglove.png
leadership/Tony-Barbagallo.jpg
1
leadership/Tony-Barbagallo.jpg
Rename
Delete
2
Metadata
Create Collection
Edit Metadata
Size 10.57 KB
Type image/jpeg



What metadata is editable?

The Content UI only exposes metadata that has been created by your organization or is otherwise appropriate for you to change. Fields that are system-controlled or generated by the request are not editable. For example, the ETag (entity tag) is a 128-bit number used to uniquely identify the object on the Internet, and it must never be altered by hand. These are the fields that are available for adding, editing and deleting:

	Examples	Exceptions	Notes
Content-*	Content-Language Content-Location Content-Type	Content-Length Content-MD5	Use care when updating required metadata, such as Content-Type. If you try to delete required metadata, Swarm will restore the last known value.
Castor-*	Castor-OLD-Metadata	Castor-System-*	These custom fields use a deprecated format, but they might still appear on older objects.
X-*-Meta	X-Country-Meta X-Zip-Postal-Code-Meta X-lat-long-Meta		
X-*-Meta-*	X-Zer0-Meta-code X-Zer0-Meta-name X-Zer0-Meta-division		

Creating metadata

When you select the Add (+) command to create new metadata, the Content UI will ensure that your field name is valid for custom metadata in Swarm. To name your metadata, follow these rules:

- Use letters with any mix of case.
- Only use numbers and dashes (-) following letters.
- Create unique field names (no duplicates allowed)

The case that you enter is preserved; `x-foo-meta` and `X-Foo-Meta` are the same field.

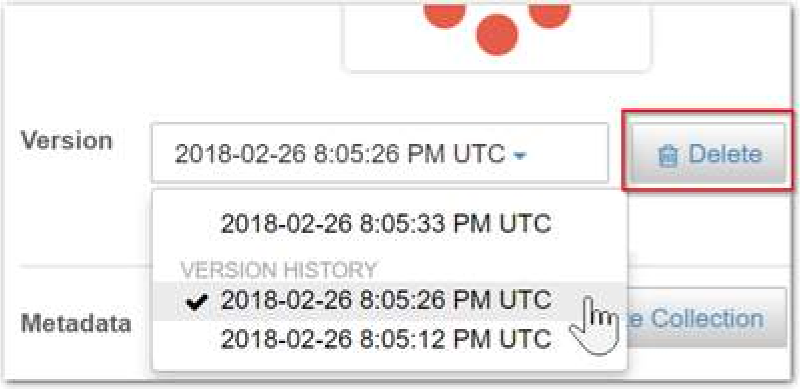
Viewing and Deleting Versions

Although you cannot edit metadata on historical versions (because they must not be altered in any way), you have the ability to delete them. The Content UI supports precise deletions of one or all of the historical versions of a given versioned object. (v5.4.0)

Viewing — To view and access the object's version history, click on the drop-down arrow in the Version list. When you select a prior version of the object, the preview image above updates for that version. Click on the thumbnail to open the historical version in another tab. You can download it from there, if needed.

Deleting — When you select a Delete button for a versioned object, you are prompted to verify which kind of delete to perform:

- Delete the object, which writes a delete marker as the new current version. Deleted objects can be restored, if needed.
- Delete the current version, which removes the specifically selected version from the version history. The deleted version cannot be restored.
- Delete all versions, which removes the current version and the entire version history. These deletions cannot be restored.


Version
2018-02-26 8:05:26 PM UTC
Delete
2018-02-26 8:05:33 PM UTC
VERSION HISTORY
✓ 2018-02-26 8:05:26 PM UTC
2018-02-26 8:05:12 PM UTC
Metadata
e Collection

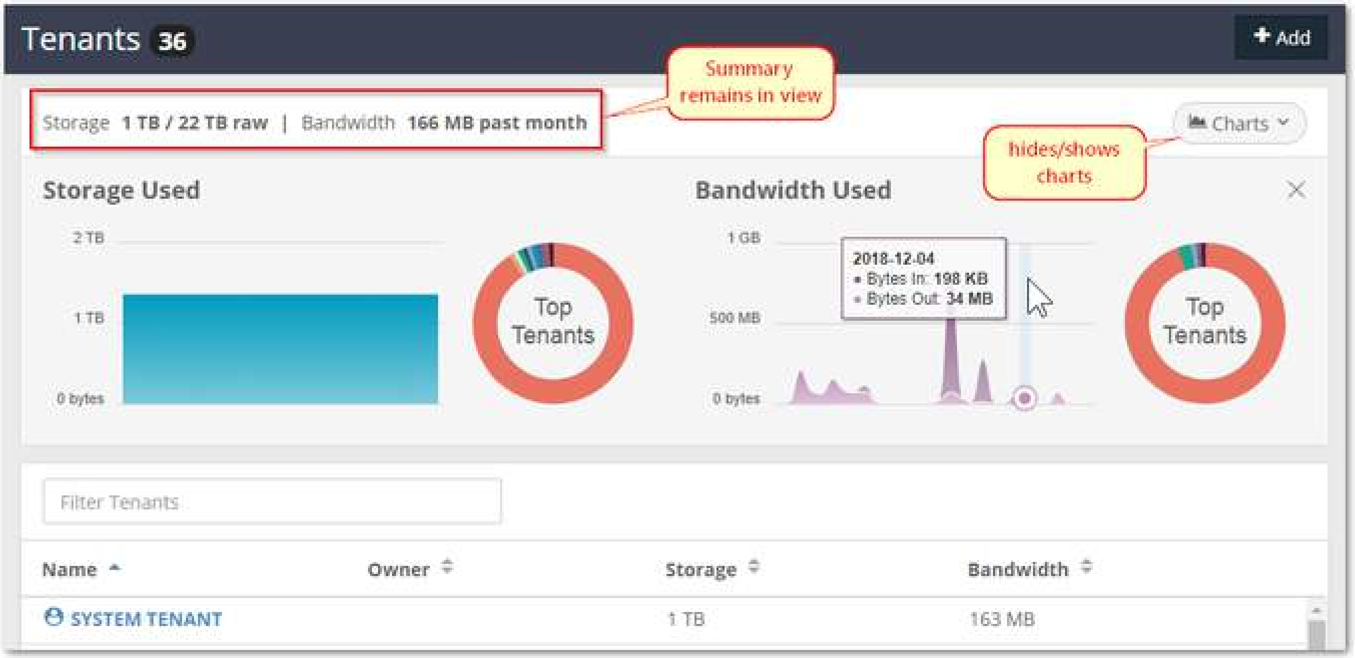
Usage Reports

- Storage Reports
- Bandwidth Reports

Each level of your Swarm site (tenants, domains, and buckets) includes a summary bar directly under the black title bar. This dynamic reporting and charting of bandwidth and storage usage is based on the data that is captured by Content Metering. Historical usage queries populate these graphs, which make it easy for you to monitor usage status visually, from the dashboard.

Tip

Hover over any data point on a chart to see a pop-up of its details.



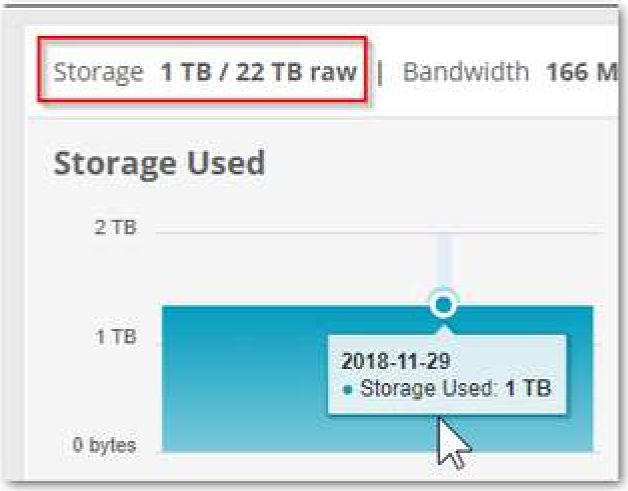
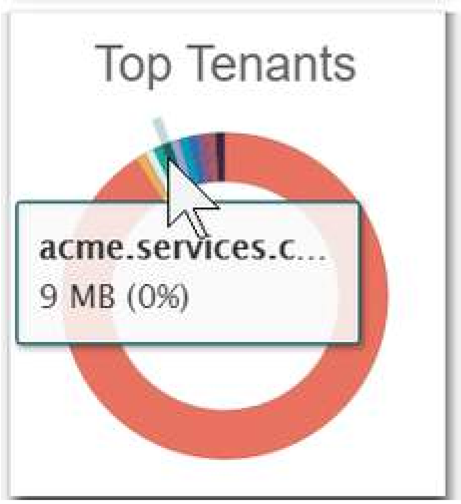
The summary bar alerts you to the Storage Used and Bandwidth Used status for the immediate context, so it is always visible. The Charts button expands and collapses the view of the bandwidth and storage usage charts that update dynamically for the tenant, domain, or bucket that is currently selected. Everything relates to the specific context being accessed; for example, if a user is allowed access to only a single bucket, she can see the storage and bandwidth status of her bucket, but only that bucket. Only those authorized to see the top-level Tenants view can see the report of the raw storage space that remains available in the cluster.

Errors - If the usage charts and totals are missing, it can have one of these causes:

- Metering is not enabled.
- Metering is enabled but without connection (which may be temporary).
- Metering data has not yet been collected.

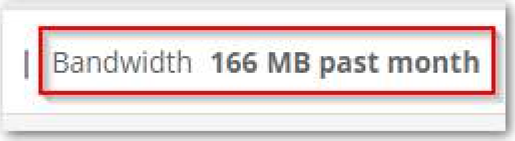
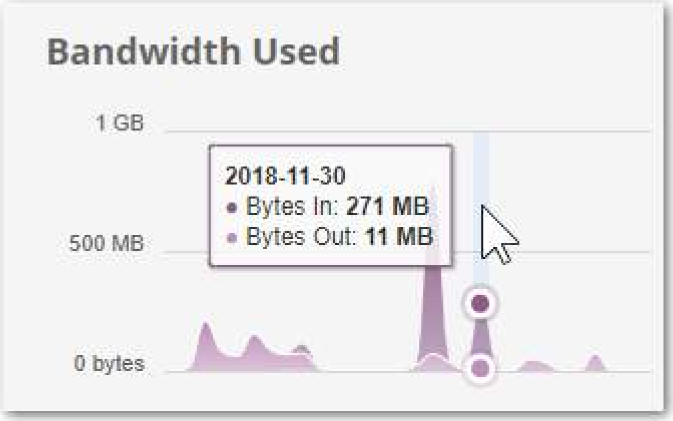
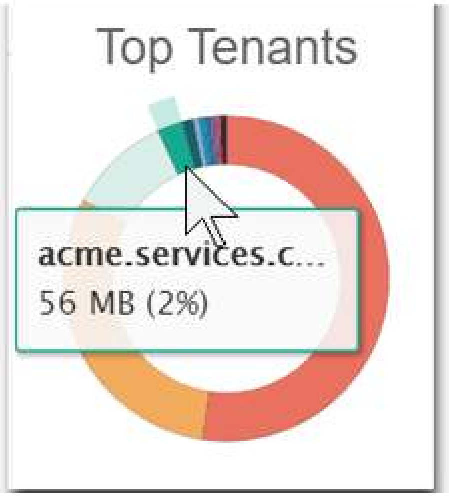
Storage Reports

The Storage reports show the amount of capacity. They are inclusive of all versions, replicas and erasure-coded segments that are stored at each level.

 Storage 1 TB / 22 TB raw Bandwidth 166 M Storage Used 2 TB 1 TB 0 bytes 2018-11-29 Storage Used: 1 TB	
Graph - A historical storage capacity report where the average of all storage data points on a given day is plotted over a rolling 30 day window. Used - The total amount of storage capacity in use for the given level. Root level only. The amount of storage capacity still available for use in the cluster.	Graph - The top 10 highest consumers of storage capacity in the given level at the time the chart was displayed. Percentage - The relative size of the displayed segment represents the percentage of the total for which a given consumer accounts. For example, if one tenant uses half of the capacity in the cluster, that tenant would be represented on the root level Top Tenants report as half of the circle.

Bandwidth Reports

The Bandwidth reports show the amount of activity. They are inclusive of all bandwidth (both bytes in and bytes out) used at each level.

 Bandwidth 166 MB past month 	
Graph - A historical bandwidth report where the sum of all bandwidth data points on a given day is plotted over a rolling 30 day window. Bytes in and bytes out are represented by two different stacked colors in the chart. Used - The total amount of bandwidth used for the given level in the last 30 days.	Graph - The top 10 highest consumers of bandwidth in the given level over a rolling 30 day window. If there are more than 10 data points for a level, the 11th and all other data points are grouped into an Other segment. Percentage - The relative size of the displayed segment represents the percentage of the total for which a given consumer accounts. For example, if one tenant uses half of the bandwidth total for the tenant over a 30 day period, that tenant would be represented on the root level Top Tenants report as half of the circle.

Note

Non-bucket content (tenanted unnamed objects) are associated with the bucket "Content IDs".

> See Content Metering.

Swarm Development

Caringo Swarm

- Storage SCSP Development

Storage SCSP Development

This section describes how to develop applications using the Caringo Swarm Simple Content Storage Protocol (SCSP), the mechanism that applications use to communicate with Swarm. SCSP is a simple, text-based protocol based on the HyperText Transfer Protocol (HTTP) 1.1 standard. Using this section, you can:

- Create an application that connects to a storage cluster.
- Implement advanced HTTP features for improved performance.

- Map SCSP methods to HTTP methods.
- Implement erasure coding in large or unknown length objects.
- Implement SCSP methods, such as READ, WRITE, and DELETE.
- Manually create or rename a domain when your cluster administrator is not available.

- Development Guidance
- Connecting to a Swarm Cluster
- SCSP Essentials
- SCSP Methods
- Search Queries
- Metadata Headers
- Content Integrity
- Multipart Write
- Working with Policies
- Replication
- Erasure Coding EC
- Object Versioning
- Time of Last Access - atime

Development Guidance

This section highlights best practices and common pitfalls for developing application integrations with Swarm. Review these sections, and be sure to develop your understanding of Swarm objects and processes.

> See Swarm Concepts.

- Application Best Practices
- Common Integration Problems

Application Best Practices

- Use the Software Development Kit (SDK)
- Protect Data in Transit
- Use Multithreading
- Maintain One Open Connection

The following are important concepts and approaches for building optimal integrations to Swarm.

Use the Software Development Kit (SDK)

The Swarm Software Development Kit (SDK) simplifies integration with Swarm by providing client library support that includes handling for specific SCSP behaviors to programmers developing Swarm applications. It describes a consistent set of features using a common API in each of several popular programming languages.

> See the SDK Overview.

Protect Data in Transit

The Content-MD5 metadata header provides an end-to-end message content integrity check (excluding metadata) of an object as it is sent and returned from Swarm.

A client application can:

- Check this header to detect modification of the object's body in transit.
- Provide this header to have Swarm compute and check it when storing or returning the data.

If a Content-MD5 header is present on POST, Swarm computes an MD5 digest during data transfer and then compares the computed digest to the one provided in the header. If the hashes do not match, Swarm returns a 400 Bad Request error response, abandons the object, and closes the client connection.

Content-MD5 headers are stored with the object metadata and returned on all subsequent GET or HEAD requests. If a Content-MD5 header is included with a GET request, Swarm computes the hash as the bytes are read. If the computed and provided hashes do not match, the connection is closed before the last bytes are transmitted, which is the standard method to indicate something went wrong with the transfer.

The Content-MD5 header provides an extra level of insurance, protecting against potential damage in transit as well as from damage while in storage.

> See Configuring the Nodes for configuration parameters and how to edit the configuration files.
> See Content-MD5 Checksums.
> See Lifepoint Metadata Headers for more information about lifecycle management.
> See Content Integrity Assurance.

Use Multithreading

Swarm is a multithreaded, multi-node cluster, which means that every node in a storage cluster can establish and maintain connections with many different client applications at the same time. Normally, an application opens only one SCSP connection to the cluster and sends requests and receive responses in a sequential manner. However, a single client application might choose to open more than one connection to Swarm to achieve better response times and read or write throughput for high-volume applications.

Because Swarm automatically load balances requests by causing them to be redirected to a less busy node in the cluster that is capable of servicing the request, this multithreaded client strategy can be very effective in improving overall performance when necessary, because each client thread (or process) can be connected to different nodes within the cluster.

Maintain One Open Connection

The Swarm software implements HTTP/1.1 persistent connections. That means a client application is not required to close the socket or connection after each request. Swarm will hold connections open and allow the client to continue sending requests and receiving responses until either the client closes the connection explicitly, or it stops sending requests for some period of time.

However, the client must close its connection and reopen a new connection whenever a Swarm response includes the header Connection: close . Typically, this is done when there is an error that would cause confusion as to the meaning of the remaining bytes sent over the connection.

Have your client maintain only one open connection at a time using one of these methods:

- Close the old connection before opening the new, redirected connection.
- Maintain a pool of connections to several nodes in the cluster. For smaller clusters, the pool approach can considerably improve response times because the client eventually has open connections to all the nodes in the cluster.

> Caution
>
> For very large storage clusters, the client must take care not to exceed the operating system limits on the number of simultaneously open connections.

Common Integration Problems

- Stale Cache in a Large System
- Workarounds for Redirect Issues

Stale Cache in a Large System

Because Swarm is a distributed system, any given client request can be directed to any node in the cluster. In a large distributed system, propagating actions takes time. As a result, certain types of transient errors can result after executing an SCSP "change" method, such as COPY, APPEND, UPDATE, and DELETE. (This also applies to WRITE, which is not considered a change method.)

For example, after you UPDATE a bucket's metadata, an INFO request for that bucket might return the old metadata. The error stops after about twice the value of the cache.realmStaleTimeout configuration parameter. (By default, `cache.realmStaleTimeout` is set to 10 minutes. After about twice that time — 20 minutes — following the APPEND, the bucket is "seen" by all nodes in the cluster.)

Tasks that include these transient errors include:

- Using SCSP UPDATE to replace a bucket's metadata.
- Deleting a bucket and creating a new bucket with the same name.
- Using SCSP COPY to add custom metadata to a bucket or domain.

After about twice the value of the `cache.realmStaleTimeout` configuration parameter occurs, the same task should succeed.

> Important
>
> These examples apply to buckets and domains and not to named and unnamed objects. Before you delete a bucket, delete or move the objects it contains.

Workarounds for Redirect Issues

Some HTTP client libraries and frameworks do not handle redirects correctly for WRITE requests, at least not by default. Older versions of the HTTP protocol (namely HTTP 1.0) were lax in their specification of exactly what the client must do when it receives a 301 or a 307 response code. Because of this, older clients, including many web browsers, developed their own conventions. While some of these conventions might be useful, they are in direct violation of the HTTP/1.1 specification and therefore incompatible with Swarm.

Both the Microsoft .NET framework and the libCURL framework (and probably others) are known to process a redirect from a POST request by changing the POST to a GET and then sending the new request to the redirect server. Workarounds exist in the impacted frameworks because the behavior is known to be in violation of the specification.

According to RFC 2616: "When automatically redirecting a POST request after receiving a 301 status code, some existing HTTP 1.0 user agents will erroneously change it into a GET request."

Connecting to a Swarm Cluster

This section describes how your application can connect to a storage cluster or node.

Requests to store, retrieve, or delete objects in the cluster can initially be sent to any accessible node. The cluster decides which node is best suited to carry out the request, based on resource availability and other factors.

Primary access node (PAN)

The node that initially fielded the request is called the PAN.

Secondary access node (SAN)

If the PAN did not field the request, it sends the application a redirect request that includes the address of a node referred to as the SAN. Using this method, a storage cluster performs automatic and intrinsic load balancing.

Any node in a storage cluster can serve as a PAN. The PAN for a particular request is the first node in the cluster that receives the request from the application. After a PAN receives a request, it decides whether to service the request itself or to request the application to redirect it to one of the other nodes in the cluster.

Because the PAN assumes responsibility for having the application direct the request to another node in the cluster, your application does not need to provide any load balancing solution. Swarm implements load balancing, even when nodes are dynamically added or removed from the cluster.

- Choosing How to Access a PAN

Choosing How to Access a PAN

Because any node can be called on by the PAN to service any particular request, all nodes must be accessible to the application client. An application can locate a PAN to use for transactions using one of the following methods.

To locate a PAN, use one of these methods (listed most preferred to least):

- Use the Swarm SDK
- Use Multicast-DNS (mDNS)
- Use DNS round robin
- Use a pool of static IP addresses
- Use a single static IP address

Use the Swarm SDK

(recommended) You can integrate your applications with Swarm using the Software Development Kit (SDK). Along with the convenience it provides, your application can use the ProxyLocator or StaticLocator object included with the SDK to locate and communicate with a node.

The SDK's ProxyLocator subclass performs two functions:

- Performs a GET / to the SCSP Proxy to pre-populate its local list of Swarm node IP addresses.
- Dynamically maintains this list as redirects and other responses are received directly from Swarm nodes.

> See the SDK Overview.

Use Multicast-DNS (mDNS)

Another way to make your nodes locate an initial PAN is to use mDNS. mDNS is often referred to as Zeroconf, the collective name for DNS and DNS Service Discovery to enable zero-configuration networking.

mDNS is supported for all deployments. It provides the most flexibility because it presents applications with a list of storage nodes to choose from when selecting a PAN without requiring the application to maintain a static list of IP addresses.

Every Swarm node implements an mDNS service that allows applications to provide service discovery. Even if DHCP is used to assign and change node IP addresses, mDNS allows an application to "discover" an active node in any storage cluster and use it as the PAN. Several free mDNS client implementations in various languages are available online for implementing mDNS node location.

> Important
>
> When using mDNS, ensure that the `cluster.name` parameter value is unique for each cluster. The parameter is located in the `node.cfg` file or in the Platform Server's cluster configuration.

Swarm mDNS support allows an application to discover all nodes on a network, all nodes in a specific cluster, or to look up a node. To implement this process, it "publishes" several different records, including an A (host) record for the node and an SRV (service) record under the _scsp._tcp service type.

Although an in-depth description of mDNS deployment is beyond this scope, a typical use example is provided below. This example uses the Avahi command line tools to pass in the name of the cluster and return all nodes discovered in that cluster. Here, two nodes were found and their IP addresses were returned in the address field for each record.

```
% avahi-browse -tr
_clustername._sub._scsp._tcp local + eth0 IPv4
D2024267FF8F1DD056EEA15E40EE52C9
_scsp._tcp local = eth0 IPv4 CD35B28FD2E70CD1E47095C774F8050F
_scsp._tcp local hostname = [CD35B28FD2E70CD1E47095C774F8050F.local]
address = [192.168.1.123] port = [80] txt = [] = eth0 IPv4
D2024267FF8F1DD056EEA15E40EE52C9
_scsp._tcp local hostname = [D2024267FF8F1DD056EEA15E40EE52C9.local]
address = [192.168.1.125] port = [80] txt = []
```

Use DNS round robin

For large and/or dynamic storage clusters where nodes are often added and removed (even for temporary maintenance), you can address the cluster using a DNS host name instead of an IP address.

This method is recommended for all deployments. It is particularly helpful for multi-tenancy, as you can use the DNS name to pass in a domain.

When using DNS with multi-tenancy, the domains must resolve to least one IP address (such as an "A" record) for client applications so that the application software includes a recognized Swarm domain name in the Host header of the HTTP/1.1 request.

> Tip
>
> With some DNS servers, you can move the maintenance of the PAN addresses out of the applications and into the DNS server itself. The Berkeley Internet Name Domain (BIND), the most commonly used DNS server, lets you enter multiple "A" records that map a single DNS name to more than one IP address.

This process also requires static IP addresses, but it enables the application to use a single DNS name (or multiple

DNS names if you are using multiple domains) for the entire cluster. The DNS server selects one of the defined IP addresses on a round-robin basis. If one of the nodes does not respond, the application must resolve the host name again.

Use a pool of static IP addresses

A less desirable approach for an application to address a storage cluster is to use a stored list of several (perhaps all) of the static IP addresses for the nodes in the cluster. This method is not recommended for a production environment.

The application's stored list of IP addresses must be accessible programmatically from the application. If one of the nodes fails to respond to a request, the application can simply try another IP address.

If a redirect response reveals a storage node that is not in the original list, the application should be able to add the new IP address to the list. If your cluster is relatively stable with respect to static node IP addresses, this may be a good approach. However, if nodes are frequently added and removed from the cluster, do not use this method.

Use a single static IP address

The simplest but least recommended (and least supported) way for an application to address a storage cluster is to assign a static IP address to at least one of the cluster nodes and then use that IP address in every request. This method should be used only in a development environment. It can be set up quickly, but is not maintainable in a larger system.

The simplicity of this approach is balanced by a significant disadvantage. If the sole PAN is taken out of service or fails for any reason, the application cannot send requests to the cluster, even though other nodes might still be functioning and all desired content is still available.

SCSP Essentials

- SCSP as a subset of HTTP
- Mapping SCSP to HTTP methods
- SCSP protocol

This overview of the Simple Content Storage Protocol (SCSP) methods explains how they map to the corresponding HTTP methods.

SCSP as a subset of HTTP

The mechanism that applications use to communicate with Swarm is a simple, text-based protocol based on HTTP. Known as the Simple Content Storage Protocol (SCSP), its methods and syntax are a proper subset of the HTTP/1.1 standard.

Although many of the optional parts of HTTP/1.1 are not implemented in SCSP (which is why the protocol is referred to as simple), all required protocol components are implemented, as well as several of the common methods.

Swarm assumes communication with an HTTP/1.1 compliant client application.

> See Working with Large Objects.
>
> See the SDK Overview for the API-level implementation of SCSP. The SDK helps developers write integrations to Swarm. The SDK includes sample code in Java, Python, C++, and C#.

Mapping SCSP to HTTP methods

The following table maps SCSP methods to their complementary HTTP methods.

SCSP Method	HTTP Method	RFC 7231 Section
READ	GET	4.3.1
INFO	HEAD	4.3.2
WRITE	POST	4.3.3
UPDATE	PUT	4.3.4
DELETE	DELETE	4.3.5
n/a	~~CONNECT~~	4.3.6
n/a	~~OPTIONS~~	4.3.7
n/a	~~TRACE~~	4.3.8
APPEND		
COPY		

SCSP protocol

Most HTTP communication is initiated by a client application and consists of a request to be applied to an object on a Swarm server. In the simplest case, this is done using a single connection between the client application and the Swarm server. Being HTTP-based, SCSP protocol consists of HTTP requests and responses:

- Requests are generated by a Swarm client (that is, any HTTP/1.1 client), with these components:
  - Request method, with URI and protocol version
  - Case-insensitive query arguments
  - Required and optional headers
- Responses are generated by one or more nodes in a storage cluster, with these components:
  - Status line, with the message's protocol version and a success or error code
  - MIME-like message, with server information, entity metadata, and possible entity-body content

> See the HTTP/1.1 specification for the semantics and nuances of HTTP.

- Formatting SCSP Commands
- SCSP Headers
- HTTP Response Codes
- Error Response Headers
- Undefined Responses from Swarm
- SCSP Query Arguments
- SCSP Compatibility and Support

Formatting SCSP Commands

- When to include domain and Host
- Calling named objects
- Calling unnamed objects

> Important
> All commands include specific formats for named objects and for unnamed objects.

When to include domain and Host

The only time domain is required is for an SCSP method on a domain object itself. Neither domain nor Host is required for requests within the default cluster domain; otherwise, the domain name must be passed as the Host in the request. (Your cluster should have one domain with the same name as the cluster, which sets up a default cluster domain.)

Client applications most often send the domain name as the Host in the request. When the Host header does not match the domain name, the client can supply the domain argument to explicitly override any value from the Host request header. A domain argument always has precedence over the Host header in the HTTP/1.1 request.

Calling named objects

The named object format is:

```
METHOD /bucketname/objectname[?query-arguments] HTTP/1.1
```

where

- bucketname is a simple, URL-encoded identifier that cannot contain slash characters (or any other character not allowed in HTTP URLs)
- objectname is any legitimate URL, which can contain slash characters

Calling unnamed objects

The unnamed object format is:

```
METHOD /[uuid][?query-arguments] HTTP/1.1
```

You specify the UUID with all SCSP methods except WRITE, in which case the cluster will return the UUID in the response if the write is successful.

> Important
> When writing unnamed objects, use a HOST header equivalent to the cluster name, the host IP address, or a domain=clusterName query arg on all requests even if you are not using domains for other purposes.

> Caution
>
> When writing unnamed objects, ensure that your application is not passing a HOST header that is neither an IP address nor a domain that exists in the cluster (unless the host header matches the cluster name). Swarm will attempt to look up the non-existent domain on every request and will wait for multiple retries before the lookup times out, impacting performance.

SCSP Headers

- Standard HTTP Headers
- Swarm-Specific Headers

The following tables list all of the headers supported or used by Swarm. Unless otherwise noted, the headers described apply to buckets, named objects, and aliased objects.

- Methods are the SCSP methods to which the header applies.
- W - Writeable (~~W~~ - Not Writeable)
- Rq - Request
- Rs - Response
- P - Persisted indicates whether the header is persisted with the object. Note that some persisted headers are not writable.

> Important
>
> The total length of all persisted metadata (keys and values combined) is limited to 32 KB. Metadata that exceeds this returns a 400 (Bad Request) error.

Standard HTTP Headers

Header	RFC Section	Methods	Rq	Rs	W	P	Description
Accept	7231 5.3.2						Not used by Swarm Storage.
Accept-Charset	7231 5.3.3		X				Specifies which character encodings (charsets) are acceptable for the response. Used in the console to resolve client requests.
Accept-Encoding	7231 5.3.4		X				Implemented in accordance with RFC-2616.
Accept-Language	7231 5.3.5		X				Specifies which natural languages are acceptable for the response. Used in the console to resolve client requests.
Accept-Ranges	7233 2.3						Not used by Swarm Storage.

Age	7234 5.1	GET, HEAD		X			The age in seconds of an item read from cache. Also sent on the status page (Age: 0). Returned when an object is served from the Swarm content cache. If Age is absent, indicates the object was retrieved from disk.
Allow	7231 7.4.1	All	X	X	W	P	For a POST request on an object, restricts subsequent access to the object. For a GET request, returns the list of allowed methods for that object. Important Allow headers have no effect on automatic deletes specified in Lifepoint headers. For best protection from deletes, use `deletable=no` lifepoints. Using lifepoints lets you block recursive deletes when a bucket or domain is deleted, causing Swarm to log a CRITICAL error that non-deletable content is present.
Authorization	7235 4.2		X				Answer an authorization challenge. Usually a repeated request after receiving a 401 (Unauthorized).
Cache-Control	7234 5.2	GET, HEAD, POST	X	X	W	P	Caching-related header. Specifies directives that must be obeyed by all caching mechanisms along the request/response chain. Values include max-age, no-cache, and no-cache-context. Returned as metadata. Exactly matches what was sent with the object on POST. Serves as both a request header and a persisted header, which Specifies whether readers of the object will access the object from cache.
Connection	7230 6.1	All	X	X			An action to perform on the connection. Includes "close", to request the client close the connection. The Swarm software implements HTTP/1.1 persistent connections, which means a client application is not required to close the socket/connection after each request. However, this header may also be absent.
Content-Encoding	7231 3.1.2.2	All		X	W	P	Activates decoding behavior. Uses registered decoders for decoding.

Content-Language	7231 3.1.3.2	All			W	P	Uninterpreted. Specifies the language(s) of the entity.
Content-Length	7230 3.3.2	All	X	X	W	P	The exact number of bytes (possibly zero) comprising the content that is contained in the message body. Exception: If you add ?checkIntegrity to HEAD and GET requests for the same object, you would see different Content-Length values in the responses. This occurs because the HEAD response returns the Content-Length of the manifest rather than the object. This is a standard HTTP header used on all requests. Swarm does not update Content-Length on a COPY, and it fails any COPY requests with a Content-Length > 0. Do not send this header on a COPY unless it has the value 0.
Content-Location	7231 3.1.4.2				W	P	Not used by Swarm Storage.

Content-MD5	2616 14.15	POST, PUT		X	W	P	Deprecated. An MD5 hash of the content of the object. Can be set on a write operation or request it be computed. This value might be synthesized for some GET requests if it is not part of the persisted headers. This header provides an end-to-end message integrity check of the content (excluding metadata) as it is sent to and returned from Swarm. A proxy or client can check this header to detect accidental modification of the entity-body in transit. Also, a client can provide this header to indicate that Swarm should compute and check it as it is storing or returning the object data. Swarm fails a COPY request with Content-MD5 if the object is stored erasure-coded and the Content-MD5 wasn't stored in the original POST or PUT. Do not add Content-MD5 with a COPY to an EC object; include it in your COPY for both erasure-coded and non-erasure-coded objects if it already existed on the object. S3 compatibility The Swarm setting scsp.autoContentMD5Computation improves S3 compatibility by automating Content-MD5 hashing, which means that you do not need to include the gencontentmd5 query argument or the deprecated Expect: Content-MD5 header on writes (although you may want to supply your own Content-MD5 header for content integrity checking). This setting is ignored wherever it is invalid, such as on a multipart initiate/complete or an EC APPEND. (v9.1)
Content-Range	7233 4.2			X			Sent with a partial entity-body to specify where in the full entity-body the partial body should be applied. Appears on read range responses to indicate the actual ranges returned.

Content-Type	7231 3.1.1.5	All		X	W	P	Media type as specified in the corresponding POST or PUT request. The value should be a valid media type registered with the IANA, but Swarm does not verify this or make assumptions about the content type or structure. This response can include other headers that contain meta-information supplied by the application that stored the content. In addition to a persisted content type, this value may appear on read range responses to indicate a multi-part response. Note `Content-type: application/castorcontext` specifies that the object is a context (domain or bucket). If the setting scsp.requireExplicitContextCreate is enabled (recommended), Swarm will not create a context object unless it includes the required header, which protects against erroneous context creation. (v9.1)
Cookie	2109 4.3.4						Not used by Swarm Storage.
Date	7231 7.1.1.2	All	X	X			The HTTP time of the message. The current date/time on the Swarm node at the time of the request.
Destination	2518 9.3						Not used by Swarm Storage.
ETag	7231 2.3	All		X	~~W~~	P	Caching-related header. The ETag (entity tag) of the specific variant of the object. The value is a double-quoted UUID. The ETag of an immutable unnamed object never changes during the entire lifecycle of the object, whereas alias object ETags change each time the object is mutated by a PUT. When Versioning is enabled, the ETag identifies the version of the object. Not writable: Although it is persisted, this header cannot be supplied on any non-admin requests.
Expect	7231 5.1.1		X				Indicates that particular server behaviors are required by the client. Values include 100-continue, Content-MD5 (deprecated; see Content-MD5 Checksums), Entity-Length (deprecated). (v9.2)

Expires	7234 5.3	GET, HEAD		X	W	P	Caching-related header. Specifies the date/time after which the response is considered stale, for caching purposes. Uninterpreted. Returned as metadata. Matches what was sent with the object on POST.
From	7231 5.5.2						Not used by Swarm Storage.
Host	7230 5.4						
If-Match	7232 3.1	All	X				Caching-related header. Used with a method to make it conditional.
If-Modified-Since	7232 3.3	GET, HEAD	X				Caching-related header. Cache coherency headers. Per the RFC, Swarm makes no attempt to enforce "If-Modified-Since" on DELETE, PUT, or COPY requests. (v9.2)
If-None-Match	7232 3.2	All	X				Caching-related header. Cache coherency headers. Note that If-None-Match:* can erroneously report that an object exists during the time window after it is flagged for deletion by policy but before it is removed from disk. This window is determined by the HP cycle time.
If-Range	7233 3.2	GET	X				Caching-related header. Cache coherency headers.
If-Unmodified-Since	7232 3.4	GET, PUT, DELETE	X				Caching-related header. Cache coherency headers.
Last-Modified	7232 2.2	All but POST		X	~~W~~	P	Caching-related header. Exactly the same as Castor-System-Created. Not writable: Although it is persisted, this header cannot be supplied on any non-admin requests.
Link	5988 5						Not used by Swarm Storage.
Location	7231 7.1.2			X			Values indicate how to access one or more replicas of the object directly. May be multi-valued indicating the locations of multiple new replicas.
Max-Forwards	7231 5.1.2						Not used by Swarm Storage.
Pragma	7234 5.4						Not used by Swarm Storage.
Proxy-Authenticate	7235 4.3						Not used by Swarm Storage.
Proxy-Authorization	7235 4.4						Not used by Swarm Storage.

Range	7233 3.1		X				Indicates that a range of data is requested.
Referer	7231 5.5.2						Not used by Swarm Storage.
Retry-After	7231 7.1.3						Not used by Swarm Storage.
Server	7231 7.4.2	All		X			Swarm software version running on the responding node. The server name and version. CAStor Cluster/{version}.
Set-Cookie	2109 4.2.2						Not used by Swarm Storage.
TE	7230 4.3						Not used by Swarm Storage.
Trailer	7230 4.4		X	X			Indicates that a trailer will be sent during chunked transfer encoding.
Transfer-Encoding: chunked	7230 3.3.1	POST, PUT, APPEND	X	X			Standard header that indicates a large object to be sent to the cluster using chunked transfer encoding. Indicates that the data is being sent with an alternate transfer encoding. Values include "chunked" and "bundle". The latter is used internally for FVR of small objects (non-Standard).
Upgrade	7230 6.7						Not used by Swarm Storage.
User-Agent	7231 5.5.3		X				Standard HTTP header for a client to identify itself.
Vary	7231 7.1.4						Not used by Swarm Storage.
Via	7230 5.7.1						Not used by Swarm Storage.
Warning	7234 5.5						Not used by Swarm Storage.
WWW-Authenticate	7235 4.1			X			Indicates an authentication challenge. Usually associated with a 401 (Unauthorized) response.

Swarm-Specific Headers

Header	Methods	Rq	Rs	W	P	Description
Authentication-Info		X				A dictionary of authentication information.
Castor-{anything}				W	P	{anything} cannot start with "system". Uninterp for client customization.
Castor-Authorization		X		W	X	Content-Level authorization.
Castor-System-Accessed	GET, HEAD		X			If disk.atimeEnabled is true, Swarm tracks the t access on read requests, which is stored and r header and is indexed as the search field 'acce

Castor-System-Alias	GET, HEAD, DELETE			₩	X	The alias UUID of an alias object, bucket, or do UUID cannot be used in any SCSP method on a bucket, or named object. Use it to execute SCS an aliased object or to delete an unnamed imm
Castor-System-Auth	SEND	X				Special headers for a GET/retrieve request. The `ssword` for an administrative account on the re only if it differs from the source cluster.
Castor-System-Bytes-Used			X			With bucket list du, the number of bytes in the replicas. Replaces Castor-Bytes-Used, which is (v9.2)
Castor-System-Bytes-Used-With-Reps			X			With bucket list du / withreps requests, the nun the query with replicas. Replaces Castor-Bytes-Used-With-Reps, which is depreca
Castor-System-CID	GET, HEAD, DELETE			₩	X	For named and tenanted unnamed objects, the the owning context.
Castor-System-Cluster	GET, HEAD, DELETE, SEND			₩	P	Name of the cluster where the object was creat updated. For SEND, the value of the `cluster.r` the destination cluster.
Castor-System-CompositeMD5	GET, HEAD, POST			₩	P	Multipart-written objects only. This composite with the completed object, is computed on the complete as the sum of the Content-MD5 value parts. This header value is remembered on sub operations but removed on other updates (PUT
Castor-System-Created	GET, HEAD, DELETE		X	₩	P	A timestamp that specifies when the object wa last updated. Uses HTTP time, an ASCII format GMT.
Castor-System-Decorates	GET, HEAD, PUT, POST, COPY	X	X	W	P	Set to the ETag of a Swarm object to be decora (annotated). When the decorated object is deleted or overwr object with this header will be reclaimed by the processor.
Castor-System-Domain	GET, HEAD		X	₩	P	Named objects only. Shows the name of the do object was created. The domain name in which tenanted. Computed on GET and HEAD. Return
Castor-System-EnforceTenancy			X			True or False depending on cluster.enforceTen response header on the status page.

Castor-System-Error-Code	All		X			Sent with any error response. The request error applicable). This code is usually a 4xx or 5xx H code. (v9.1)
Castor-System-Error-Text	All		X			Sent with any error response. The request error applicable). Provides a description of the error.
Castor-System-Error-Token	All		X			Sent with any error response. A unique error to applicable). Provides an easily parsed token th identifies the error. (v9.1)
Castor-System-Headers-Filtered	GET, HEAD		X			Indicates that the response headers have been whitelist or blacklist. See Filtering Headers. (v9
Castor-System-IsVersioned	GET, HEAD			₩	P	Header that captures the object versioning stat effect at the time the object was written. Appea updateable (alias or named) objects, and only i was enabled (true) or suspended (false) at the was written. If versioning was disabled for the header does not appear. To check this value with a search query, add th uery argument. (v10.0)
Castor-System-LicenseCapacityTB			X			Return on a cluster status request. Indicates th capacity, in terabytes.
Castor-System-LicenseSerialNumber			X			Returned on a cluster status request. Indicates serial number, or 'unregistered.'
Castor-System-Name	GET, HEAD, DELETE			₩	P	Named objects only. Symbolic name of the obj the user-Specified (partial) name for this objec
Castor-System-Next-Version			X			Appear only on admin or administrative GET or responses. When object versioning is enabled named and alias objects will provide these hea if there is a previous or next version in the vers the object.
Castor-System-Object-Count			X			In a bucket listing request, the number of objec the query. Replaces Castor-Object-Count, which (v9.2)
Castor-System-Owner	All			₩	P	The user id of the authenticated user who wrot Name of the user who created or last modified the object was created anonymously (that is, w authenticating), this header is absent.

Castor-System-PartNumber		X			P	Special headers for multipart write requests. Id part number of a part in the temporary manifes write.
Castor-System-Path	GET, HEAD		X	₩	P	For named objects only, shows the full path of /domain/bucket/object format. Example: `/clu` `.com/mybucket/mypath/myobject.html`. Co and HEAD. Returns {domain}/{bucket}/{name}.
Castor-System-Previous-Version			X			
Castor-System-RecursiveDelete					P	Header on a context delete marker with a times which content therein should be deleted. The s "now" may also be used for HP to delete the co immediately.
Castor-System-Source		X				Special headers for a GET/retrieve request. Co list of IP/name:port.
Castor-System-Sources		X				Special headers for a GET/retrieve request.
Castor-System-Stream		X				Special headers for a GET/retrieve request.
Castor-System-Target	SEND	X				Special headers for a GET/retrieve request. The node or reverse proxy of the destination cluste
Castor-System-TotalGBAvailable			X			A response header on the status page.
Castor-System-TotalGBCapacity			X			A response header on the status page.
Castor-System-UploadID		X			P	Special header for multipart write requests. As header, it identifies the multipart write request subsequent requests relating to the upload. As header, it associates a stream with a multipart as the init stream or a part, based on its Castor-System-PartNumber value.
Castor-System-Version	All			₩	P	A numerical version number associated with m This is a Unix-style floating-point time since G millisecond accuracy. The timestamp (in seco epoch) when the object was written, which cor Last-Modified. Not writable: Although it is pers header cannot be supplied on any non-admin r

Composite-Content-MD5	POST	X		W		Provides an end-to-end integrity check of the c (excluding metadata) of a parallel write reques completion. The supplied value is a base-64 en concatenation of the binary Content-MD5 hash (in order) followed by a dash, followed by the n as text value. On the complete request, Swarm the value for the overall request and reject the 409 error if the values do not match.
Content-UUID	POST, COPY, PUT		X	~~W~~	P	One header per UUID specified. Indicates the al UUID from a diskless countrep HEAD request, a write request indicating the new content UUID. objects only. Remember that COPY and PUT ar methods for immutable objects. Not writable: A persisted, this header cannot be supplied on ar requests.
Entity-MD5	POST, PUT, COPY, APPEND	X	X			MD5 hash of the object's disk representation. U replica transfer. Swarm internal use only. For content verificatic Content-MD5 Metadata Header.
Feed-{id}-Status			X			Only for an administrative or admin request. Or HEAD, return the status of each defined feed, o The {id} is the feed id. The value is 0 for succes other value, the number of attempts made to pu
Feed-{id}-StatusTime			X			Only for an administrative or admin request. Or HEAD, return the status time of each defined fe header. Time is HTTP time.
Keep-Alive			X			Sent to clients as an indication that the connec stay open forever.
Lifepoint	GET, HEAD		X	W	P	Time-based SCSP and HP directives. Returned that was created with a lifepoint header. Erasure coding Swarm returns a 400 (Bad Request) resp requests with Lifepoints with EC specific regardless of whether the object already same Lifepoint set on it or is currently st same EC spec values. This means that a requests must remove all Lifepoints with

Manifest	All	X	X			Indicates the object is erasure coded. Indicates type of the object being created or queried. Inte replica transfer. Currently, "ec" is the only value Indicates the manifest type of the object being queried.
Node-Status			X			Returned on a cluster status request. Indicates node status.
Overlay-IPs			X			Up to two IP headers may appear. A list of spac addresses associated with the overlay key. Onl countreps HEAD request.
Overlay-Key			X			An overlay index key used for this object. Only HEAD request. Up to two keys may appear.
Policy-*	COPY			W	P	For COPY, include any and all Policy-* headers inclusion list.
Policy-{Feature}-Evaluated	GET, HEAD		X	₩		For all policy features (replication, EC encoding reports the value of the current policy evaluatic context objects only; to view it for all objects, a e" query argument .
Policy-{Feature}-Evaluated-Constrained	GET, HEAD		X	₩		For all policy features (replication, EC encoding reports what existing constraints are affecting evaluation. If "no", nothing constrains the polic If "yes", something (anchor, conflict, override) i policy at this level. Other values (such as "min> constraints in force. Appears for context objec it for all objects, add the "verbose" query argun
Policy-ECEncoding				W	P	Optional; context objects only. Stores the enco for erasure coding named objects in this conte unspecified, disabled, k:p (a tuple such as 5:2 t the data and parity encoding to use). For a dom "anchored" cancels any policies on its buckets.
Policy-ECEncoding-Unnamed				W	P	Optional; domain objects only. Stores the enco for erasure coding unnamed objects tenanted i Valid values: unspecified, disabled, k:p (a tuple that specifies the data and parity encoding to u
Policy-ECMinStreamSize				W	P	Optional; context objects only. Stores the polic minimum object size that triggers erasure codi context (domain, bucket). Values include "1MB minimum), "nMB", "nGB".

Policy-ECMinStreamSize-Unnamed				W	P	Optional; domain objects only. Stores the minir size that triggers erasure coding of unnamed o in this domain. In units of megabytes (MB) or g must be 1MB (default) or greater.
Policy-Replicas				W	P	Optional; context objects only. Stores the replic how to replicate objects this context (domain, l include "min:n", "max:n", "default:n", and option on domains.
Policy-Versioning				W	P	Optional; context objects only. Stores the objec licy for how to version objects this context (dor Values include "disabled", "enabled", "suspende "required".
Replica-Count	HEAD		X	~~W~~	P	Reports the number of replicas found with a co Returns the number of replicas created on a RC write. One header per UUID with a diskless cou Specifies the number of known replicas of the c cluster. Is returned only if the countreps=yes qu is included in the request. Not writable: Althou persisted, this header cannot be supplied on ar requests.
ScspHoldBucket			X			The SCSP hold bucket on a hold request.
ScspHoldDomain			X			The SCSP hold domain on a hold request
Volume	POST, PUT, COPY, APPEND	X	X		P	Indicates the volume UUID where one or more r stored. Used with replica transfer. Specifies the volume to which the object was created or moc replicate-on-write request, there are two sets o Volume headers. Not writable: Although it is pe header cannot be supplied on any non-admin re Indicates the volume UUID where one or more r stored. The Volume headers generated usually the Location headers generated.

X-{anything}-Meta[-{anything}]	POST, PUT, COPY	X		W	P	Caringo custom metadata header. Uninterprete client customization. Custom metadata has the X-*-Meta[-*], such as X-color-Meta or X-phonehome-Meta-Castor-cluster-id. The nam case-insensitive (consistent with the HTTP/1.1 Custom metadata that does not match this forı X-foo) will not be persisted. Important For COPY operations, if you omit any exi: headers, you are removing that metadata object. Add the `preserve` query argumeı COPY request to ensure that any custom existing on the object is carried over to t overwrite an existing value, include the h with the new value on the request. (v9.2)
X-Castor-Copy-Source X-Castor-Copy-Source-Domain X-Castor-Copy-Source-If-Match X-Castor-Copy-Source-If-Modified-Since X-Castor-Copy-Source-If-None-Match X-Castor-Copy-Source-If-unmodified-Since X-Castor-Copy-Source-Range		X				Special headers for a parallel part upload as a (Conditional headers evaluate the same as the c headers. X-Castor-Copy-Source is required; the optional. See Uploading the Parts.
X-Castor-Meta-Error-Message	All		X			Deprecated; replaced by Castor-System-Error-C Castor-System-Error-Text, and Castor-System-E (v9.1) Sent on a GET or HEAD response with any erroı request error description (if applicable).
X-Timestamp			X			On a bucket list request, the value of the Castor-System-Created header.

Filtering Headers

If you use Swarm to deliver content directly over the Internet, you might want to enable filtering of the optional HTTP response headers that are transmitted for GET and HEAD requests. (v9.5)

Because the header filtering does add additional processing to Swarms responses, best practice is to enable it only for a specific content delivery need:

- You need to conserve bandwidth and want to eliminate as many bytes as possible when serving content.
- You need enhanced security and want to reveal as little as possible about your content and its context.

You should not filter headers if your client applications are object storage aware and are using SCSP or S3 (Content

Gateway) to interact with Storage. Filtering metadata headers on objects may cause problems for applications that know how to work with object metadata. Examples of applications that use object metadata are the Content Gateway's Content UI, SwarmNFS, and S3 applications.

> Important
>
> Regardless of filtering, never expose Swarm Storage directly on the Internet. Do not allow arbitrary requests, especially by unauthorized users.

Header filtering is a Storage feature that you can implement dynamically (without a cluster restart). You have your choice of filtering approaches:

- Whitelist — list which non-required headers to retain, if any
- Blacklist — list which non-required headers to remove, preserving all others

The lists are case-insensitive, and they can include system headers (such as "`Castor-System-Owner`").

> Essential headers
>
> The following essential metadata headers are unaffected by Blacklisting and will always be included when they are present on an object:
>
> Allow, Authentication-Info, Authorization, Cache-Control, Connection, Content-Length, Content-MD5, Content-Range, Content-Type, Date, Expires, Keep-Alive, Location, Server, Trailer, Transfer-Encoding

Settings for Filtering

Filtering is disabled by default. These SCSP settings let you control which of the optional response headers are returned from your cluster:

scsp.filterResponseHeaders	none	Which method to use to filter HTTP response headers. Whitelist or blacklist setting must be defined before implementing that method. Valid values: none, whitelist, blacklist. SNMP: filterResponseHeaders
scsp.filterResponseBlacklist	[]	Which headers to remove from HTTP GET and HEAD responses. List is comma-separated and case-insensitive. SNMP: filterResponseBlacklist
scsp.filterResponseWhitelist	[]	Which headers to retain in HTTP GET and HEAD responses, removing all others. List is comma-separated and case-insensitive. Leave the brackets empty to have Swarm strip out all non-essential headers. SNMP: filterResponseWhitelist

> Best practice
>
> To avoid a window when filtering is enabled but your filter list is empty, define the whitelist or blacklist first an d then enable filtering by setting `scsp.filterResponseHeaders.`

You set these values using the Storage UI, or you can use SNMP or cURL:

```
curl -i
http://$SCSP_HOST:91/api/storage/clusters/<cluster-name>/settings/scsp.f
ilterResponseWhitelist
 -XPUT -d {"value": ["key1","key2"]}

curl -i
http://$SCSP_HOST:91/api/storage/clusters/<cluster-name>/settings/scsp.f
ilterResponseHeaders
 -XPUT -d {"value": "whitelist"}
```

Sample Output

Following are examples of how responses can appear with and without filtering applied. Swarm includes the `Castor-System-Headers-Filtered: True` header with every response that has been filtered by a whitelist or blacklist.

Target of GET	Headers Not Filtered	Headers Filtered

Missing Object	$ curl -i
	"172.16.15.180/11111111111111111111111111111111"
	HTTP/1.1 404 Not Found
	Castor-System-Error-Token: NotFound3
	Castor-System-Error-Text: Existing object not found in cluster.
	Castor-System-Error-Code: 404
	Castor-System-Cluster: CAStorCluster
	Content-Length: 83
	Content-Type: text/html
	Date: Fri, 30 Nov 2018 16:27:36 GMT
	Server: CAStor Cluster/9.6.a
	Allow: HEAD, COPY, GET, SEND, PATCH, PUT, RELEASE, POST,
	HOLD, GEN, APPEND, DELETE
	Keep-Alive: timeout=14400
	`<html><body><h2>CAStor Error</h2><br>`<br>Requested stream was not found`</body></html>`
		$ curl -i
		"172.16.15.179/1111111111111111111111
		HTTP/1.1 404 Not Found
		Castor-System-Headers-Filtered: True
		Content-Length: 83
		Content-Type: text/html
		Date: Fri, 30 Nov 2018 16:29:22 GMT
		Server: CAStor Cluster/9.6.singleip
		Allow: HEAD, COPY, GET, SEND, PATCH, PUT POST,
		HOLD, GEN, APPEND, DELETE
		Keep-Alive: timeout=14400
		`<html><body><h2>CAStor Error</h2><br>`<br>Requested stream was not found`</body><`

Immutable Object	$ curl -i "172.16.15.178/7b9a25bcd48afac3156a89212859c62c" HTTP/1.1 200 OK Castor-System-Cluster: CAStorCluster Castor-System-Created: Fri, 30 Nov 2018 16:31:04 GMT Content-Length: 0 Last-Modified: Fri, 30 Nov 2018 16:31:04 GMT Etag: "7b9a25bcd48afac3156a89212859c62c" Volume: b9ec90023e27941147b3ce6fb2ed54bd Date: Fri, 30 Nov 2018 16:32:13 GMT Server: CAStor Cluster/9.6.a Keep-Alive: timeout=14400	$ curl -i "172.16.15.179/7b9a25bcd48afac3156a892 HTTP/1.1 200 OK Content-Length: 0 Castor-System-Headers-Filtered: True Date: Fri, 30 Nov 2018 16:31:25 GMT Server: CAStor Cluster/9.6.singleip Keep-Alive: timeout=14400
Named Object	$ curl -i "172.16.15.180/bucket/stream?domain=domain" -I HTTP/1.1 200 OK Castor-System-CID: 84c1cbf7d33aec1feec4d4dd11225b87 Castor-System-Cluster: CAStorCluster Castor-System-Created: Fri, 30 Nov 2018 16:33:44 GMT Castor-System-Name: stream Castor-System-Version: 1543595624.202 Content-Length: 0 Last-Modified: Fri, 30 Nov 2018 16:33:44 GMT Etag: "46ce386cdc13828d7d8d68ee20aac58d" Castor-System-Path: /domain/bucket/stream Castor-System-Domain: domain Volume: 0a9a7ed07b5f86520b096fb0ef824846 Date: Fri, 30 Nov 2018 16:34:21 GMT Server: CAStor Cluster/9.6.a Keep-Alive: timeout=14400	$ curl -i "172.16.15.179/bucket/s?domain=x" HTTP/1.1 200 OK Content-Length: 0 Castor-System-Headers-Filtered: True Date: Fri, 30 Nov 2018 16:33:48 GMT Server: CAStor Cluster/9.6.singleip Keep-Alive: timeout=14400

HTTP Response Codes

Following are HTTP response codes that you may receive from Swarm, with notes about any Swarm-specific meaning.

> See RFC 7231 section 6 of the HTTP/1.1 specifications for information about response status codes.

Response	Methods	Notes
100 Continue	POST, PUT, APPEND	See SCSP WRITE, WRITE for Large Files (Expect: 100-continue). See SCSP Compatibility and Support, Issues with 100-Continue Header
101 Switching Protocols	-	Informs the client about the server switching the protocols to the one specified in the Upgrade message header field during the current connection.

200 OK	GET, HEAD, DELETE	Standard response for successful requests. For EC, indicates that Swarm found enough segments to recreate the object, which is a prerequisite for success.
201 Created	POST, PUT, COPY, APPEND	The success response for a POST or PUT request.
202 Accepted	POST	Request accepted (such as for a multipart completion), but not yet processed.
203 Non-Authoritative Information	-	Returned meta information was not the definitive set from the origin server.
204 No Content	-	Request succeeded without requiring the return of an entity-body.
205 Reset Content	-	Request succeeded but requires resetting of the document view that caused the request.
206 Partial Content	GET	Successful response to a GET that includes one or more Range headers, returning the specific range data.
207 Multi-Status	-	
300 Multiple Choices	-	Requested resource has multiple choices at different locations.
301 Moved Permanently	All	Resource permanently moved to a different URL Requests the client to resend the current request to the location supplied in the response headers and to direct all future requests to that new node until further notice. The `location` header supplies the authorization parameter to be included in the request to the new PAN.
302 Found	-	Requested resource was found under a different URL but the client should continue to use the original URL.
303 See Other	-	Requested response is at a different URL and can be accessed only through a GET command.

304 Not Modified	GET, HEAD, PUT, APPEND, COPY	If-Modified-Since condition not met. Requested object was not modified since the last request. Requested object was not modified since the time specified in the If-Modified-Since header. The response is returned without any message-body. If-None-Match condition not met on GET or HEAD.
305 Use Proxy	-	Requested resource should be accessed through the proxy specified in the location field.
307 Temporary Redirect	All	Resource has been moved temporarily to a different URL. The client should resend the current request to the location supplied in the response headers, but to continue using the original PAN for the next request until further notice. See Application Best Practices, Use Workarounds for Redirects.
400 Bad Request	All	
401 Unauthorized	All	Occurs as the normal initial response of HTTP authentication for a domain. Administrative request lacks an Authorization header with suitable administrative credentials. The response includes a WWW-Authenticate challenge containing the administrative domain named Castor administrator and other required items.
403 Forbidden	Varies	Unsupported method was used. Alias objects only support POST, DELETE, GET, and HEAD. Domain objects only support GET, HEAD, COPY, PUT, and APPEND.
404 Not Found	GET, HEAD, APPEND, PUT, COPY, DELETE	Indicates that the content could not be located in this cluster. Common causes include these: Object deletion Request errors (such as the wrong bucket name) Network failure Node(s) down for maintenance Timeouts due to a heavily loaded or extremely active cluster Rebalancing due to new drive capacity in the cluster Requesting aliased objects without using the etag flag, because the overlay index does not keep primary UUIDs for aliased objects. Appending to an immutable object or a UUID that does not exist. Multipart range index out of range. Attempting to read a version not associated with this alias.

405 Method Not Allowed	-	Method specified in the Request-Line was not allowed for the specified resource. APPEND not allowed for content-encoded objects. Allow header forbids this method on this object.
406 Not Acceptable	-	Resource requested generates response entities that has content characteristics not specified in the accept headers. Content-encoding not acceptable, or not found on request or in 'decoderSettings' setting.
407 Proxy Authentication Required	-	Request requires the authentication with the proxy.
408 Request Timeout	-	Client fails to send a request in the time allowed by the server.
409 Conflict	PUT, POST, APPEND, COPY, DELETE	Request was unsuccessful due to a conflict in the state of the resource. Includes the following conditions: Attempting to create a domain or bucket object that already exists. Renaming a domain or bucket to a name that already exists. Attempting to update an immutable object. Attempting to update an alias object in a domain that does not exist. Rapid updates of an object, resulting in the error "Later version already exists." Persisted Content-MD5 did not match value on request. Basis object is no longer erasure-coded. Encoding has changed on the basis object since the initiate. Initialized object must have matching uploadID, which might have been updated since the initiate.
410 Gone	GET, HEAD, DELETE	Delete marker. Erasure-coded object: too few segments found to service request, checksum failure trying to generate an EC segment, or unable to read sufficient objects to complete request.
411 Length Required	-	Server cannot accept the request without a valid Content-Length header field. Content-Length must be provided for all non-EC object requests. Content-Length must be provided and must be zero for COPY request.

412 Precondition Failed	All	Precondition specified in the Request-Header field returns false. For erasure coding, indicates that Swarm could not recreate the object, listing the missing segments in the body of the response. For named objects, indicates that the named object already exists or that the bucket or domain cannot be found. For unnamed objects, indicates Swarm did not write the object because the domain does not exist, if cluster.enforceTenancy is set to true. See WRITE for Unnamed Objects. For replication, indicates that the cluster cannot locate at least two nodes to initially store the replicas. Check the Replica-Count header to verify that Swarm created the correct number of replicas. When Swarm initiates a replication request to a PAN and the replication or initial write fails, Swarm fails both procedures and generates a 412. This guarantees that two copies of the object are saved on separate nodes in the cluster before returning a 201 response.
413 Request Entity Too Large	-	Server is not ready to receive the large file and has closed the connection. See WRITE for Large Files (Expect: 100-continue).
414 Request-URI Too Long	-	Request unsuccessful because the URL specified is longer than the server can process.
415 Unsupported Media Type	-	Request unsuccessful because the entity of the request is in a format not supported by the requested resource.
416 Requested Range not satisfiable	GET	GET request includes invalid Range headers because the request is out of bounds of the data. Invalid range header, could not parse. x-castor-copy-source-range was not satisfiable. Cannot start past the end of the content, and start must be less than the end.
417 Expectation Failed	-	Expectation given in the Expect request-header was not fulfilled by the server. gencontentmd5 query argument or Expect: Content-MD5 header (deprecated) is not supported for APPEND. gencontentmd5 query argument or Expect: Content-MD5 header (deprecated) is not allowed on multipart write initiate request. List operations do not support any 'Expect' headers.

500 Internal Server Error	All	Critical error in Swarm. Check your logs for more information and contact Support if necessary. Tip A 500 error is almost always accompanied by a non-500 'child' error with more detail, which is returned in the error headers.
501 Not Implemented (Forbidden Feature)	-	Requested method is unsupported in Swarm. Only the methods listed in the Allow header currently work in Swarm. Erasure coding is not enabled in the configuration. List operations unavailable because Elasticsearch is not configured or licensed. List operations require a Search Feed, but none are currently available. Destination server version is invalid or does not support remote replication. Destination server is not recognized, or destination server name is not provided. Destination cluster.name is not set, or differs from expected.
502 Bad Gateway	-	Server received an invalid response from the upstream server while trying to fulfill the request. Could not find a source for RETRIEVE operation. Castor-System-Cluster header has an invalid cluster name.
503 Service Unavailable (Try Again)	All	Processing was interrupted, so the client should try again. Common causes include the following: • Swarm could not complete due to a transient problem or inadequate resources to process the request. • Cluster is too busy to service additional requests. • Gateway cannot communicate with Swarm.
504 Gateway Timeout	-	Upstream server failed to send a request in the time allowed by the server. Cannot connect to destination.
505 HTTP Version not supported	All	Indicates that a request was received with an HTTP version other than HTTP/1.1. Swarm only supports HTTP/1.1.
507 Insufficient Storage Space	PUT, POST, APPEND, COPY	Request cannot be completed because of space limitations or licensing restrictions/errors.

Error Response Headers

SCSP error responses have headers detailing the error code, response code (token), and error description. The response for any request with an error code 400 or greater includes three special headers: `castor-system-error-co` de, castor-system-error-token, and castor-system-error-text. These headers replace the legacy header x-castor-meta-error-message, which is deprecated. (v9.1)

Tip

A failure response may contain a series of cascading errors. Focus on the final error in the series, which is likely to be the issue that you need to address:

```
< HTTP/1.1 412 Precondition Failed
< Castor-System-Error-Token: SecurityRealmFailure
< Castor-System-Error-Text: Failed to load context
'some-domain.example.com/nosuchbucket'.
< Castor-System-Error-Code: 404
< Castor-System-Error-Token: RequiresContext2
< Castor-System-Error-Text: Cannot find required domain or bucket.
< Castor-System-Error-Code: 412
< Content-Length: 130
< Content-Type: text/html
...
```

- Code values of "0" indicate that the response code is not static and will be populated by Swarm when the error is generated.
- Text values of "{0}" or "{1}" indicate a variable that will be populated by Swarm when the value is generated

Code	Token	Text
0	CloseException	{0}
0	CloseException2	{0}
0	CloseException3	{0}
0	CompletionErrorNonRequest	{0}
0	CompletionErrorRequest	{0}
0	NotFoundDeleted	Requested object was not found because it has been deleted.
0	NotFoundDeleted3	Requested object was deleted.
0	NotFoundDeleted5	Requested object was deleted.
0	NotFoundDeletedExisting	Existing object found deleted in the cluster.
0	NotFoundExistingDeleted	Existing object found deleted in the cluster.
0	NotFoundWrite2	Requested object was deleted.

0	ReaderIndexError2	Index error.
0	ReaderInvalidRead3	Unexpected destination cluster response.
0	ReaderNotFoundAlreadyDeleted	Requested object was not found, already deleted.
0	ReaderProxyError	Failed to read object necessary to proxy.
0	ReaderRequiresMarker	Failed to create delete marker in cluster.
0	ReaderRequiresRead	Unexpected read error attempting to proxy object.
0	ReaderSegmentError	Unable to read a segment.
0	ReaderSegmentError2	Unable to info a segment.
0	ReaderSegmentError3	Unable to read a segment.
0	ReaderSegmentError7	Unable to info segment.
0	SecurityRealmFailure	Failed to load context '{0}'.
0	SecurityStreamFailure	Failed to load object.
0	UnexpectedException	Unexpected exception.
0	UnexpectedHoldFailure	Unexpected failure trying to create default HOLD domain.
0	UnexpectedHoldFailure2	Unexpected failure trying to create default HOLD _administrators object.
0	WriterInvalidExtentArg	extentsize' must be an integer.
0	WriterInvalidHold7	Unable to INFO bucket during finalize.
0	WriterInvalidHold8	Unable to finalize HOLD bucket.
0	WriterInvalidInfo	Failed initiating info of part.
0	WriterParallelDeleteFail	Failed to delete init manifest.
0	WriterPartReadFailure	Unable to read part.
0	WriterProxyFailure	Replication peer request failed.

0	WriterRequiresDestination	Replication peer failed to specify destination volume.
0	WriterRequiresExtentsize	extentSize' argument is required for chunked transfer encoding.
304	ConditionalIfModified	If-Modified-Since condition not met.
304	ConditionalIfNoneMatch	If-None-Match condition not met on GET or HEAD.
400	CopyMD5Mismatch	The Content-MD5 provided on the COPY request does not match the value in the manifest.
400	DigesterForbiddenArgs	Content Integrity: 'hashtype' and 'newhashtype' queryArgs cannot be used together.
400	DigesterInvalidHashtype	Content Integrity: unsupported hash type.
400	DigesterMismatch	Persisted {0} did not match request.
400	DigesterMismatch2	{0} did not match computed digest.
400	DigesterMismatch3	Local Content-MD5 did not match remote Content-MD5.
400	DigesterMismatch4	{0} did not match computed digest.
400	DigesterMissingHeader	Expect {0} trailing header not supplied.
400	DigesterRequiresHash	Content Integrity - 'hash' query arg required with 'hashtype' on GET or COPY.
400	DigesterRequiresHashtype	Content Integrity - 'hashtype' query arg required on request.
400	ForbiddenAction	Action' query arg is not allowed on existing object request.
400	ForbiddenAliasUUID	Only the COPY operation supports renaming using aliasuuid.
400	ForbiddenConditional	Conditional headers other than If-None-Match:* on a named request not allowed in a POST.
400	ForbiddenContext	Cannot write duplicate context.
400	ForbiddenEtag	Etag query argument not appropriate on write requests.
400	ForbiddenGenID	IsGenId query arg must be used with GET or HEAD methods only.
400	ForbiddenIndexWaitValue	Forbidden value '{0}' with index query argument

400	ForbiddenManifest	{0} not appropriate on this request.
400	ForbiddenManifestEC	Cannot provide erasure coding query args for a manifest write.
400	ForbiddenManifestHeader	{0}' header only allowed on a POST.
400	ForbiddenManifestHeader2	{0}' header not allowed on a context request.
400	ForbiddenPolicyHeader	Policy headers are only allowed on domains and buckets.
400	ForbiddenPutCreate	Putcreate' query argument not allowed on context requests.
400	ForbiddenSegmentEC	Cannot EC an existing segment.
400	ForbiddenSegmentedEC	Cannot specify segmented=yes, and provide erasure coding query args.
400	ForbiddenSegmentSize	Cannot specify 'segmentsize' on an existing EC object.
400	ForbiddenSpec	Cannot specify 'erasurecoded' on an existing object for given method.
400	ForbiddenSpec2	Cannot specify 'encoding' on an existing object for given method.
400	ForbiddenSpec3	Cannot specify 'segmentwidth' on an existing object for given method.
400	ForbiddenSpec4	Cannot specify 'segmentsize' on an existing object for given method.
400	ForbiddenSpec5	Cannot specify 'lifepoint k:p' on an existing object for given method.
400	ForbiddenStreamHeader	Duplicate header values detected for {0}.
400	ForbiddenVersioning	Version' query argument may not be used on a context request.
400	ForbiddenVersioning2	Version' query argument may not be used on an immutable request.
400	ForbiddenVersioning3	DEPRECATED: 'Version' query argument not appropriate in current state.
400	InvalidAliasUUID	Aliasuuid' must be a UUID.
400	InvalidAuthorization	CAStor-authorization header error.
400	InvalidBucketName	Illegal character in bucket name.

400	InvalidCID	Cid' queryArg must be valid UUID.
400	InvalidContentLength	Content-Length must be zero for COPY request.
400	InvalidContentLength1	WritePattern query argument requires a message body with contentLength greater than 0.
400	InvalidContentLength2	WritePattern pattern must not be larger than value of scsp.writePatternMax.
400	InvalidContentMD5	Content-MD5 not allowed on multipart write initiate request.
400	InvalidContentMD52	Content-MD5 value can not be blank.
400	InvalidContentMD53	Content-MD5 value was not not a valid base64 md5 hash.
400	InvalidCount	Count' query arg must be > 0 and <= scsp.maxreplicas.
400	InvalidCount2	Count' query arg must be an integer > 0 and <= scsp.maxreplicas.
400	InvalidDecorates	{0} header may not be added to contexts.
400	InvalidDecorates2	{0} header must be a single UUID.
400	InvalidDecorates3	{0} header must be a valid UUID.
400	InvalidDecorates4	{0} header value must refer to an ETag.
400	InvalidDomainName	Illegal character in domain name.
400	InvalidDomainsDomain	Request must not provide both 'domains' and 'domain' query arguments.
400	InvalidDomainSpecified	Domain may not change on a recreatecid request.
400	InvalidECCombination	Cannot specify no encoding, and provide erasure coding query args.
400	InvalidECEncoding	Invalid EC encoding.
400	InvalidECQueryArgs	Error parsing EC queryArgs.
400	InvalidEntityLength	Expected integer for 'Entity-Length' header value.
400	InvalidEntityLength2	Entity length header does must match Content-Length header.

400	InvalidMethod	A PATCH request is only valid on a multipart upload initiate.
400	InvalidModifiedSince	If-Modified-Since time in the future.
400	InvalidNewName	Domain rename name contains invalid character.
400	InvalidNewName2	Bucket or object rename name is invalid.
400	InvalidObjectName	Object names cannot look like a UUID.
400	InvalidParallelEncoding	Part uploads cannot specify encoding.
400	InvalidParallelFlag	Part uploads cannot be alias objects.
400	InvalidParallelMethod	Part upload must be a POST.
400	InvalidParallelRename	Part uploads cannot specify a new name.
400	InvalidParallelRequest	Context requests cannot be made by multipart upload.
400	InvalidPartNumber	Part number must be an integer at least 1.
400	InvalidPolicyHeader	Unrecognized policy header.
400	InvalidPolicyHeader2	Unnamed policy cannot be used on a bucket.
400	InvalidPolicyValue	Policy value is not valid.
400	InvalidPolicyValue2	Policy value is malformed or invalid.
400	InvalidQueryArgCombo	The 'replace' and 'preserve' query args can not be used on the same request.
400	InvalidRangeTime	If-Range time in the future.
400	InvalidRecreateCID	The value of recreatecid must be a UUID.
400	InvalidReplicate	Replicate' query arg must be > 0 and <= scsp.maxreplicas.
400	InvalidReplicate2	Replicate' query arg must be a keyword, or an integer > 0 and <= scsp.maxreplicas.
400	InvalidRepSpec	Cannot specify 'erasurecoded' and reps=X.
400	InvalidRepSpec2	Cannot specify 'encoding' and reps=X.

400	InvalidRepSpec3	Cannot specify 'segmentwidth' and reps=X.
400	InvalidRepSpec4	Cannot specify reps=X on a chunked upload. Chunked uploads must use EC.
400	InvalidSegmentSize2	SegmentSize must be greater than or equal to value of segmented.minSegmentSize.
400	InvalidSourceHeader	{0} header value must be in the form :.
400	InvalidSourcesHeader	{0} header value must be a comma-separated list in the form :.
400	InvalidStreamHeader	Header '{0}' is not syntactically valid.
400	InvalidUnmodifiedSince	If-Unmodified-Since time in the future.
400	InvalidUploadID	UploadId query arg value was not a well-formed uploadid value.
400	InvalidURI	Invalid URI. Bucket and object name must be percent-encoded utf-8 bytes.
400	InvalidURI2	URI resource does not match request.
400	InvalidVersion	Invalid 'version' query argument value.
400	InvalidVersionContext	Version' query argument may not be used on a context request.
400	InvalidVersionMethod	Version' query argument may not be used with request method.
400	InvalidWritePattern	Value for writePattern must be a positive integer.
400	InvalidWriteRandom	Value for writeRandom must be a positive integer.
400	MetaInvalidUUID	There are no valid UUIDs to query for countreps.
400	RangeForbidden	Range not allowed on COPY or APPEND.
400	RangeInvalid	Invalid range; index greater than range.
400	ReaderBucketError	List operation specifies non-context object.
400	ReaderBucketError2	List request must not provide both 'domains' and 'domain' query arg.
400	ReaderBucketError3	List request must provide either a 'domain' or 'domains' query arg.

400	ReaderBucketError5	List request must not provide a UUID.
400	ReaderBucketError9	Indexer searches on selected field not supported.
400	ReaderInvalidArg	Invalid query argument 'scsphold' on DELETE request. Objects in an SCSP HOLD cannot be removed using the DELETE verb.
400	ReaderInvalidFields	Fewer sort fields than marker values not supported.
400	ReaderInvalidFormat	Invalid format value. Must be one of: json, xml.
400	ReaderInvalidHeader	{0} header for remote cluster must be different than cluster.name setting.
400	ReaderInvalidHeader2	{0} header value should be in the form :.
400	ReaderInvalidHeader3	{0} header value should be of the form :.
400	ReaderInvalidHeaders	Error formatting object headers.
400	ReaderInvalidHeaders2	Error decoding Castor-Hold-Meta header.
400	ReaderInvalidHeaders3	Error decoding Castor-Hold-Meta header.
400	ReaderInvalidRead	Destination cluster reports 400 on read request.
400	ReaderInvalidRecursive	Recursive' query arg must be an integer.
400	ReaderInvalidStreamResults	Invalid streamresults value. Must be one of: true, false, yes, no.
400	ReaderInvalidStreamType	Invalid stype value. Must be one of: {0}.
400	ReaderInvalidUUID	Missing or invalid UUID.
400	ReaderInvalidVersion	{0} header not allowed on non-administrative requests.
400	ReaderInvalidVersion2	{0} header must be a float.
400	ReaderInvalidVolume	Invalid volume specified.
400	ReaderInvalidVolume2	Volume specified is not ready.
400	ReaderRequiresHeader	SEND request requires {0} header.
400	ReaderRequiresHeader2	{0} header not provided on request.

400	ReaderRequiresName	cluster.name setting is required for SEND request.
400	RequiredBidLength	Either a content length header or extentsize query arg is required to create a bid.
400	RequiredLength	A request must specify a length or qualify for EC in order to estimate required license space.
400	RequiresAdmin	Domain rename requires 'admin' queryArg.
400	RequiresCluster	{0} not specified on request.
400	RequiresContext	The 'recreatecid' query arg requires that either a domain or bucket is specified on the request.
400	RequiresDecorates	{0} header object was not found.
400	RequiresDomain4	Part upload for tenanted write must include domain query arg.
400	RequiresEncoding	Multipart writes must specify a valid EC encoding in the cluster, or on the request.
400	RequiresExplicitBucket	Bucket creation requires the 'Content-type: application/castorcontext' header.
400	RequiresExplicitContext	Context creation requires the 'Content-type: application/castorcontext' header.
400	RequiresInitiated	Part upload must specify the uuid or name of the initiated object.
400	RequiresLocalCluster	{0} value must refer to the local cluster on an administrative request.
400	RequiresName	Cid' queryArg requires a named content or domain specification.
400	RequiresRecursiveQueryArg	All DELETES on a context must include the recursive query arg.
400	RequiresRemoteCluster	{0} value must refer to a remote cluster.
400	RequiresSourceHeader	{0} header not sent on request.
400	RequiresUUIDName	Missing or invalid UUID.
400	RequiresValidEncoding	Multipart writes must specify a valid EC encoding in the cluster, or on the request.
400	WriterExcessMetaData	Too much persisted metadata.

400	WriterInvalidContentLength	Content length for EC object exceeds maximum supported size ec.maxSupported.
400	WriterInvalidExtent	extentsize' must be greater than or equal to contentLength.
400	WriterInvalidManifest	Could not parse part manifest.
400	WriterInvalidPartDict	Invalid completion manifest. Parts entries must be dictionaries.
400	WriterInvalidPartDict2	Invalid completion manifest. Ranges entries must be dictionaries.
400	WriterInvalidPartId	Each part in the part manifest must contain a part number and a uuid, or a range.
400	WriterInvalidPartId2	Each part in the part manifest must contain a part number and a uuid.
400	WriterInvalidPartNumber	Part number found in part ({0}) does not match part number in completion manifest ({1}).
400	WriterInvalidPartUuid	Part uuid string was not valid UUID.
400	WriterInvalidPartUuid2	Part uuid string was not valid UUID.
400	WriterInvalidPath	x-castor-copy-source was not valid UUID.
400	WriterInvalidRange	x-castor-copy-source-range was not valid.
400	WriterInvalidReplicationPost	Content-Type: {0} is only appropriate on an authorized admin POST.
400	WriterInvalidSetCount	Total EC encoded sets will exceed maximum number of sets for this segment size. Try a larger segment size.
400	WriterInvalidSort	Allowed values for sortOrder are 'part' and 'natural.'
400	WriterInvalidType	Unknown manifest type in segment header.
400	WriterInvalidType2	Unknown manifest type in manifest header.
400	WriterInvalidUploadId	UploadID in part does not match UploadID provided in request.
400	WriterManifestBadCO	contentOffset must be a number zero or greater.
400	WriterManifestBadSize	Size must be a number zero or greater.
400	WriterManifestInvalidContentLength	Query arg 'contentLength' must be a number; at least zero.

400	WriterManifestInvalidPartOffset	Part offset must be a number of zero or greater.
400	WriterManifestInvalidPartUuid2	Could not form uuid from uuid string.
400	WriterManifestInvalidStartSize	Can not specify a size and offset that exceeds the bounds of a part, for part number {0}.
400	WriterManifestInvalidValue	Generated value must be a single character.
400	WriterManifestMissingPartNumber	Every uuid entry must have a part number.
400	WriterManifestMissingShrink	Truncating an object with PATCH requires the 'shrink' query arg.
400	WriterManifestNoContentRange	Every range must have a contentOffset specified.
400	WriterManifestNoSize	Every range must have a size specified.
400	WriterManifestOverlap	Content ranges can not overlap.
400	WriterManifestOverwrite	Can not write past specified contentLength.
400	WriterManifestUUIDAndValue	A range can not specify both a uuid and a generated value.
400	WriterMD5Mismatch	Persisted {0} did not match value on request.
400	WriterMD5Mismatch2	Persisted {0} did not match value on EC request.
400	WriterRequiresManifest	The complete manifest was not provided.
400	WriterRequiresMD5	Validating the Content-MD5 on an EC COPY requires an existing Content-MD5 stored on the object.
400	WriterRequiresPartlist	Completion manifest contained no valid parts.
400	WriterRequiresParts2	Part manifest must contain a populated 'parts' or 'ranges' key, but not both.
400	WriterRequiresPatch	Multipart upload must be initiated by PATCH to supply a range on contentLength on the completion.
400	WriterRequiresRange	Multipart upload by PATCH requires a 'range' key, and does not accept a 'parts' key on completion.
400	WriterRequiresSubclusters	Based on ec.protectionLevel=subcluster, and ec.subclusterLossTolerance, this write cannot succeed.

400	WriterRequiresUniqueParts	Duplicate part number received in completion manifest.
400	WriterSegmentOverflow	Request exceeds the segment limit.
400	WriterSegmentOverflow2	Request exceeds the segment limit.
400	WriterSizeLimit	EC encoded sets size exceeds supported size ec.maxSupported on APPEND.
400	WriterTooMuchData	Too much data written for body.
400	WriterTypeConflict	An object may not be both a segment and a manifest.
401	ReaderAuthError	Destination cluster authorization error.
401	ReaderDeleteNotAuthorized	Unauthorized administrative request.
401	ReaderUnauthorized	Unauthorized administrative request.
401	ReaderUnauthorizedInternode	Unauthorized internode request. Segments requests require admin auth.
401	VersionUnauthorized	Unauthorized administrative request.
401	WriterUnauthorized	Unauthorized internode request.
401	WriterUnauthorized2	Unauthorized internode request.
401	WriterUnauthorized3	Unauthorized internode request.
403	ForbiddenUUID	UUID forbidden on POST.
403	ForbiddenUUIDName	UUID/Name forbidden on RETRIEVE.
403	InvalidRealm	POST not allowed to this context.
403	ReaderDeleteNotDeletable	object is not deletable at this time, based on its lifepoint policy.
403	RequiresOwned	Attempted owner access to a non-owned object.
403	RequiresRealm	Content specified unknown realm.
404	NotFound3	Existing object not found in cluster.
404	NotFound4	Requested object was not found.

404	NotFound5	Requested object was not found.
404	NotFound6	Requested object was not found.
404	NotFoundDeleted2	Existing object found with delete marker.
404	NotFoundDeleted4	Requested object has a delete marker.
404	NotFoundDeleted6	Requested object has a delete marker.
404	NotFoundDeleted7	Requested object was deleted by policy.
404	NotFoundDeleted8	Requested object was deleted by policy.
404	NotFoundDeleted9	Requested object was deleted by policy.
404	NotFoundDisk	Object was not found on disk.
404	NotFoundExisting	Existing object not found in the cluster.
404	NotFoundMarker	Versioned object has a delete marker.
404	NotFoundMarker2	Object has a delete marker.
404	NotFoundNotVersioned	Requested object was not found, not versioned.
404	NotFoundNotVersioned2	Requested object was not found, not versioned.
404	NotFoundOld	Looking for an older version in a versioning disabled state.
404	NotFoundOld2	Looking for an older version in a versioning suspended state.
404	NotFoundOld3	Requested object's current version was not found in the cluster.
404	NotFoundOverlay	Requested object was not found, per overlay index lookup.
404	NotFoundOverlay2	Requested object was not found, per overlay index lookup.
404	NotFoundOverlay3	Requested object was not found, per overlay index lookup.
404	NotFoundOverlay4	Requested object was not found, per overlay index lookup.
404	NotFoundVersion	Requested object was not found in the cluster.
404	NotFoundWrite	Requested object was not found.

404	Reader404NotFound	The 404stream query arg was provided.
404	ReaderCacheDeleted2	Requested object was found with a delete marker.
404	ReaderDeleteNotFound	Object has been deleted.
404	ReaderNotFound3	Attempting to read a version not associated with this nid.
404	ReaderNotFound5	Attempting to read a version not associated with this alias.
404	ReaderNotFoundCacheDeleted	Requested object was found with a delete marker.
404	ReaderNotFoundPolicy	Destination object is not found, is deleted by policy.
404	ReaderNotFoundProxy	Primary UUID not found while proxying object.
404	VersionNotFound	Requested object was found with a delete marker.
405	EncodeInvalidMethod	APPEND not allowed for content-encoded objects.
405	ForbiddenMethod	Allow header forbids this method on this object.
406	EncodeInvalidCE	Content-encoding not acceptable.
406	EncodeRequiresCE	Content-encoding not found on request or in 'decoderSettings' setting.
409	ForbiddenContextName	POST will only create new context objects. Context already exists.
409	ForbiddenContextName2	POST will only create new context objects. Context already exists.
409	ForbiddenMutable	Operation on mutable object must be by name or alias.
409	ForbiddenMutable2	Can not PUT an immutable object.
409	ForbiddenMutable3	Can not PUT an immutable object.
409	ForbiddenMutable4	Can not COPY an immutable object.
409	ForbiddenMutable5	Can not APPEND an immutable object.
409	ForbiddenMutable6	Can not {0} an immutable object.
409	ForbiddenSegmented	Cannot segment an EC object.

409	ForbiddenSegmented2	Cannot segment an EC object.
409	GenRequiresEncoding	GEN can only be performed on segments, by administrative request.
409	InvalidParallelState	Cannot PUT a multipart write object still in progress.
409	InvalidParallelState2	Cannot COPY a multipart write object still in progress.
409	InvalidParallelState3	Cannot APPEND a multipart write object still in progress.
409	InvalidUploadID2	Initialized object must have matching uploadID. It might have been updated since the initiate.
409	NotFoundObsolete	Object version obsolete.
409	NotFoundObsolete2	Object version present or obsolete.
409	ReaderContextMismatch	Object context mismatch. Invalid domain or possible attempt to tamper with data.
409	ReaderContextMismatch2	Object context mismatch. Invalid domain or possible attempt to tamper with data.
409	ReaderContextMismatch3	Domain specified for untenanted object.
409	ReaderInvalidCID	Object context mismatch. Invalid domain or possible attempt to tamper with data.
409	ReaderInvalidCID2	Object context mismatch. Domain specified for untenanted object.
409	ReaderInvalidId	Mutable objects must be deleted by name or alias.
409	ReaderInvalidUploadId	UploadID in object was not found or does not match UploadID in request.
409	RequiresBasisGeneration	Generation of basis stream has changed since init.
409	RequiresBasisHeader	The initiate object does not refer to the request object.
409	RequiresEncoding2	Could not find encoding header in initiated object.
409	RequiresEncoding3	Encoding has changed on the basis since the initiate.
409	RequiresEncoding4	Could not find encoding header in basis object.
409	RequiresUniqueCreated	{0} must be unique.

409	RequiresUniqueVersion	{0} must be unique.
409	RequiresVersionHeader	{0} header requires {1} header.
409	RequiresVersionHeader2	{0} header requires {1} header.
409	UnexpectedExceptionMeta	Unable to load object metadata for APPEND.
409	VersionCidInvalid	Object context mismatch. Domain specified for untenanted object.
409	VersionCidMismatch	Object context mismatch. Context id on request does not match context in object.
409	VersionConflict	Unable to complete request due to conflicting resources: non-POST on a object with no primary UUID.
409	VersionUpdateError	Alias updates occurring too quickly. Make sure clocks are synced.
409	WriterBundleDuplicate	Object already exists.
409	WriterInvalidContentMD5	Calculated composite MD5 does not match provided composite MD5.
409	WriterInvalidHold3	Specified HOLD request already exists. Try using a different name.
409	WriterLocked	A request for the specified object is already in progress on this node.
409	WriterLocked2	Simultaneous internode writes of same bundle UUID.
409	WriterNotPermitted	Operation not permitted.
409	WriterNotPermitted2	Operation not permitted.
409	WriterOutOfDate	Cannot POST out-of-date version.
409	WriterParallelBasisFail	Basis object is no longer ECed.
409	WriterParallelBasisFail2	Basis object is no longer ECed.
409	WriterRequiresContentMD5	Missing required Content-MD5 on part {0}.
410	ReaderSegmentError5	Not enough segments found to service request.
410	WriterChecksumFailure	Checksum failure trying to GEN ec segment.
410	WriterRequiresStreams	Unable to read sufficient objects to complete GEN request.

411	RequiresContentLength	Content-Length not provided for non-EC request.
411	RequiresContentLength2	Content-Length must be provided and must be zero for COPY request.
412	ConditionalIfMatch	If-Match condition not met.
412	ConditionalIfMatch2	If-Match condition not met.
412	ConditionalIfNoneMatch2	If-None-Match condition not met.
412	ConditionalIfUnmodified	If-Unmodified-Since condition not met.
412	ConditionalMatchNotFound	A matching object was not found on If-Match request.
412	ConditionalNoneMatchFound	A matching object was found on If-None-Match request.
412	DigesterSealMismatch	Content Integrity - seal did not validate.
412	ForbiddenDomainName	Cannot rename domain to existing domain name.
412	ForbiddenRename	Cannot rename due to existing object.
412	InvalidBasisStream	Basis object has been deleted.
412	InvalidURI3	Could not resolve domain for context specified in CID header.
412	NotFoundInitiateDeleted	Initiated object found deleted in the cluster.
412	ReaderBucketError4	Supplied realm does not exist.
412	ReaderDeleteNoExist	Failed to open object for deletion.
412	RequiresBasisStreamNotFound	Could not find basis object.
412	RequiresContext2	Cannot find required domain or bucket.
412	RequiresDomain	Request requires a domain specification.
412	RequiresDomain2	Tenancy enforced and failed to find or load domain '{0}'. If creating a domain, include the 'Content-type: application/castorcontext' header.
412	RequiresDomain3	A loadable domain must be specified when enforceTenancy=True.
412	RequiresInitiateStream	Initiate object could not be found.

412	RequiresInitiateStream2	Initiate object has been deleted.
412	RequiresLocalCluster2	{0} value must refer to the local cluster on an administrative request.
412	WriterHeaderMismatch	{0} header does not match {1} header.
412	WriterInvalidEncoding	Invalid encoding header.
412	WriterInvalidHold	Empty HOLD request body is not allowed. Request body must specify at least one object.
412	WriterInvalidHold2	Too many items in HOLD request. Maximum is cluster.scspHoldMaxItems.
412	WriterInvalidSegment	Segment is not part of this set.
412	WriterProxyRequiresHeaders	Replication peer request failed.
412	WriterRequiresCreated	{0} header is required.
412	WriterRequiresEncoding	{0} header is required.
412	WriterRequiresExtentsize2	extentsize' query arg is required.
412	WriterRequiresGeneration	{0} header is required.
412	WriterRequiresNodes	Unable to find sufficient nodes for replication.
412	WriterRequiresRead	Unable to read part.
412	WriterRequiresRead2	Unable to read part.
412	WriterRequiresSegment	{0} header is required.
412	WriterRequiresSiblings	{0} header is required.
412	WriterRequiresSource	Could not find source object.
412	WriterRequiresUniqueCreated	Unique {0} header is required.
412	WriterTimeout	Timed out waiting for peer write.
416	RangeInvalidHeader	Invalid range header; could not parse.
416	RangeInvalidHeader2	Invalid range header; could not parse.

416	WriterInvalidRange2	x-castor-copy-source-range was not satisfiable. Cannot start past the end of the content.
416	WriterInvalidRange3	x-castor-copy-source-range was not satisfiable. Start must be less than end.
417	ForbiddenContentMD5	Expect:Content-MD5/gencontentmd5 is not supported for APPEND.
417	InvalidExpectContentMD5	Expect: Content-MD5/gencontentmd5 not allowed on multipart write initiate request.
417	ReaderBucketError6	List operations do not support any 'Expect' headers.
500	CloseFailure	{0}
500	CloseFailure2	{0}
500	CloseFailure3	{0}
500	CloseFailure4	{0}
500	CompletionErrorUnknown	Unknown completion error.
500	DigesterMissingRemote	Remote replica did not contain a Content-MD5.
500	EnvelopedHeadersTimeout	Content-Type: %s used, but enveloped headers were not sent before timeout.
500	InternodeUnexpectedCount	Found an unexpected number of nodes, fewer than needed.
500	InvalidOwner	Invalid owner header format.
500	MultiHeaderMismatch	Local and remote replicas are not identical.
500	OpenFailure	{0}
500	OpenFailure2	{0}
500	RangeUnexpectedContent	Unexpected extra content in multipart response.
500	ReaderDeleteError	Disk error on open.
500	ReaderDeleteException	Exception deleting object.
500	ReaderDifferent	Named object lookup failure.

500	ReaderInvalidManifest	Invalid manifest.
500	ReaderInvalidManifest2	Invalid manifest.
500	ReaderInvalidMetadata	Too much persisted metadata on delete.
500	ReaderInvalidType	Unexpected manifest object type.
500	ReaderMissingDigest	Partial object, has no digest.
500	ReaderNotFound6	Alias object lookup failure.
500	ReaderNotFound8	IOError opening existing object.
500	ReaderRequiresRead2	Unexpected exception when proxying for EC manifest.
500	ReaderSegmentError4	Unknown read error.
500	ReaderSegmentError6	Caught RequestError during ec segment prepare.
500	ReaderUnexpectedBatch	Batch handler exception.
500	ReaderUnexpectedJournal	Could not iterate journal.
500	ReaderUnexpectedStatus	Error computing status page.
500	ReaderUnexpectedType	Unexpected object type.
500	VersionUnexpectedError	Error opening existing anchor object.
500	WriteFailure	{0}
500	WriteFailure2	{0}
500	WriteFailure3	{0}
500	WriteFailure4	{0}
500	WriterBundleError	IOError creating bundled object.
500	WriterBundleException	Exception creating bundled disk object.
500	WriterInvalidHold4	Create HOLD bucket failed with response code {0}.
500	WriterInvalidHold5	Unable to INFO source object for HOLD request, downstream error.

500	WriterInvalidHold6	Unable to add object to HOLD request, downobject error.
500	WriterManifestIncomplete	Could not parse the manifest.
500	WriterManifestIncomplete2	Could not parse the manifest.
500	WriterMissingLocalMD5	The local replica did not compute the Content-MD5.
500	WriterMissingRemoteMD5	A remote replica did not compute the Content-MD5.
500	WriterParallelDeleteException	Unexpected exception trying to delete init manifest.
500	WriterRequiresExtentsize3	ProxyWriter requires extentSize when writing in chunked mode.
500	WriterRequiresManifest2	Unable to create manifest.
500	WriterRequiresManifest3	Unable to create manifest.
500	WriterRequiresManifest4	Unable to create manifest.
500	WriterRequiresManifest5	Unable to create manifest.
500	WriterRequiresManifest6	Unable to create manifest.
500	WriterRequiresManifest7	Unable to write manifest.
500	WriterRequiresPartManifest	Invalid part manifest.
500	WriterTooMuchData2	Object failed to generate digest.
500	WriterUnexpectedException2	Unexpected exception in StreamWrite.
501	ForbiddenFeature	You must configure Erasure Coding to obtain this functionality.
501	ForbiddenFeature2	You must configure Erasure Coding to obtain this functionality.
501	ReaderBucketError7	List operations unavailable because indexer is not configured or not licensed.
501	ReaderBucketError8	List operations require Indexing Feed, but none currently available.
502	ReaderInvalidRead4	Destination server version does not support remote replication.
502	ReaderInvalidRead5	Invalid destination server version.

502	ReaderInvalidRead6	Destination server is not recognized.
502	ReaderInvalidRead7	Destination server name is not provided.
502	ReaderNoSources	Could not find a source for RETRIEVE operation.
502	ReaderRequiresCluster	Destination cluster.name is not set.
502	ReaderRequiresClusterMatch	Destination cluster.name differs from expected.
503	InternodeInvalidState	Failed to detect valid subcluster bids. Try again.
503	InvalidBirthdate	Temporary alias conflict after upgrade: try again.
503	ReaderDeleteNoSpace	Not enough space in node.
503	ReaderDeleteOutOfMemory	Not enough index memory to complete the operation. Try again.
503	ReaderIndexError	Index error: try again.
503	ReaderNotFound2	Requested object was not found on disk: try again.
503	ReaderNotFound7	Object version found is obsolete; try again.
503	ReaderRequiresMarker2	Exception trying to create delete marker in cluster.
503	ReaderUnavailableIndex	Search index unavailable. Wait indexer.insertBatchTimeout seconds and try again or check log for indexer errors: try again.
503	ServerFinalizing	Server cannot process client requests at this time: try again.
503	ServerFinalizing2	Server cannot process client requests at this time: try again.
503	ServerFinalizing3	Server cannot process client requests at this time: try again.
503	ServerInitializing	Server cannot process client requests at this time: try again.
503	ServerInitializing2	Server cannot process client requests at this time: try again.
503	ServerInitializing3	Server cannot process client requests at this time: try again.
503	VersionDoesNotExist	Out-of-date version: try again.
503	WriterBundleNoSpace	Not enough space in node.

503	WriterOutOfMemory	Out of memory: try again.
503	WriterOutOfSpace	Not enough space in node.
503	WriterParallelObsolete	Object version found is obsolete; try again.
503	WriterParallelObsolete2	Object version found is obsolete; try again.
503	WriterParallelOpenFail	Failed to open object; try again.
503	WriterParallelOpenFail2	Failed to open object; try again.
503	WriterRequiresConnection	Unable to connect to peer.
503	WriterRequiresReplicas	Not enough replicas to succeed. Failing pipeline.
503	WriterRequiresReplicas3	No replicas were created.
503	WriterRequiresVolume	Targeted volume is not present.
503	WriterRequiresVolume2	Targeted volume is not available.
503	WriterSequenceObsolete	Object version found is obsolete; try again.
503	WriterShutdown	Node shutdown: try again.
503	WriterVolumeFailed	Volume failed, try again.
504	ReaderInvalidRead2	Cannot connect to destination.
507	ClusterOutOfObjects	Not enough licensed objects in the cluster for request.
507	ClusterOutOfSpace	Not enough licensed space in the cluster for request.
507	ClusterOutOfSpace3	Not enough space in cluster.
507	ClusterOutOfSpace4	Not enough space in cluster.
507	InternodeInvalidDistribution	Did not balance across the correct number of subclusters.
507	InternodeInvalidDistribution2	Did not balance evenly across subclusters.
507	InternodeRequiresNodes	No remote nodes are available to take objects of specified size.

507	InternodeRequiresNodes2	Not enough remote nodes are available to take objects of specified size.
507	InternodeRequiresNodes3	Not enough subclusters are available to take objects of specified size.
507	InternodeRequiresNodes4	No nodes found in bid auction.
507	InternodeRequiresSubclusters	Cannot apply subcluster protection when the segment count is not greater than the required segments per subcluster.
507	RequiresHealthReporting	Health reporting is required by license.
507	WriterRequiresReplicas2	Could not find at least two nodes to do replicate on write.

Undefined Responses from Swarm

- Critical error in Swarm
- Unsupported request
- Unavailable service
- Unsupported HTTP version

This section describes HTTP requests sent to a storage cluster where the results are currently undefined. In most cases, one of the following error responses is sent by Swarm.

Critical error in Swarm

```
HTTP/1.1 500 Internal Server Error
 Date: Wed, 1 Sept 2012 15:59:02 GMT
 Server: CAStor Cluster 5.0.0
 Allow: GET, HEAD, POST, PUT, DELETE
 Connection: close
 Content-Length: 27
 Content-Type: text/html CRLF
 Message
```

Check your logs for more information and contact your support representative if necessary.

Unsupported request

```
HTTP/1.1 501 Not Implemented
 Date: Wed, 1 Sept 2012 15:59:02 GMT
 Server: CAStor Cluster 5.0.0
 Allow: GET, HEAD, POST, PUT, DELETE
 Connection: close
 Content-Length: 56
 Content-Type: text/html CRLF
 Swarm does not understand the request or does not yet implement this
functionality.
```

This response indicates a request method was received that Swarm does not implement yet. Only the methods listed in the Allow header currently work in Swarm.

> See SCSP Headers.

Unavailable service

```
HTTP/1.1 503 Service Unavailable
 Date: Wed, 1 Sept 2012 15:59:02 GMT
 Server: CAStor Cluster 5.0.0
 Allow: GET, HEAD, POST, PUT, DELETE
 Connection: close
 Content-Length: 0
 Content-Type: text/html
```

This response indicates Swarm received a request it did not understand or it does not have the resources to process the request. The client should resubmit the request at a later time or to a different node in the cluster.

Unsupported HTTP version

```
HTTP/1.1 505 HTTP Version not supported
 Date: Wed, 1 Sept 2012 15:59:02 GMT
 Server: CAStor Cluster 5.0.0
 Allow: GET, HEAD, POST, PUT, DELETE
 Connection: close
 Content-Length: 0
 Content-Type: text/html
```

This response indicates a request was received with an HTTP version other than HTTP/1.1. Swarm only supports

HTTP/1.1.

SCSP Query Arguments

- Query Arguments by Area
- Support and Administration Arguments

To use arguments, put a question mark (`?`) after the URI, add the query argument, and put an ampersand (`&`) before any and all subsequent arguments. See RFC 3986, section 3.4.

```
http://yourURI?arg1=value&arg2=value&arg3=value
```

> See Search Query Arguments, which are specific to Swarm's Elasticsearch integration.

Query Arguments by Area

The following table describes all of the SCSP query arguments, which are grouped by feature or purpose.

- Query argument names are case-insensitive, as are most values.
- A valueless query argument (with no =) is changed internally to true.
- Write requests include POST, PUT, APPEND, and COPY.
- Read requests include GET and INFO.
- Booleans have equivalent forms: `alias, alias=yes, alias=true`

Applies to	Name	Value(s)	Description and usage
Alias Objects	alias	`yes/true` to activate	On write requests, indicates that an alias object should be created. The alias argument must be used with a POST method on an alias object and can optionally be included for other operations on alias objects.
Content Integrity	gencontentmd5	`yes/true` to activate	Computes the Content-MD5 for the body data of the request, returning the Content-MD5 as a header in the 201 Created response. Replaces the `Expect: Content-MD5` header, which is deprecated. (v9.2)
	hash	hash value	A content integrity hash value provided on the request for validation. The case of this argument does not matter.
	hashtype newhashtype	{ md5 | sha1 | sha256 | sha384 | sha512 }	Specifies the value of an object's hash. The `hashtype` query argument may appear on a variety of requests to generate or validate a content hash value. The `newhashtype` query argument is used to "re-seal" the content hash with one hash value while simultaneously checking another.

	validate	`yes/true` to activate	Validate On Read (VOR). Reads an object with an integrity seal. See Content Seals and Validation. On GET, validates that the data read from disk has not been corrupted. If this check fails, Swarm closes the connection before all of the data is sent.
Domains	createdomain	value ignored	[Deprecated: v9.2] Add to a WRITE to create the domain specified by **`domain=domain-name`**. See Manually Creating and Renaming Domains.
	domain	domain-name	Represents the domain name in some SCSP requests. Client applications most often send the domain name as the Host in the request. When the Host header does not match the domain name, the domain argument can be supplied by the client to explicitly override any value from the Host request header. A domain argument always has precedence over the Host header in the HTTP/1.1 request. The only time domain is required is for an SCSP method on a domain object itself. Neither domain nor Host is required for requests within the default cluster domain.
Erasure Coding (EC)	encoding	k`:`p	The integer values for the data (k) and parity (p) segment counts when specifying erasure coding.
	erasurecoded	`yes/true` to override, `no/false` to inhibit	Used on EC writes to override the cluster's **`policy.ecMinStreamSize`** value. ● YES forces EC encoding for objects smaller than the cluster's minimum. ● NO prevents EC encoding on objects that might otherwise be erasure-coded. Using **`erasurecoded`** without **`encoding`** or when the cluster is not configured for EC will result in a 400 Bad Request.
	segmentsize	integer	The maximum size (in bytes) of a segment in any erasure-encoded set for this object, overriding the **`ec.SegmentSize`** configuration setting. This value cannot be smaller than 100 MB.
	segmentwidth	integer	Number of bytes. Allows **`ec.segmentWidth`** configuration value to be modified per request.
Listing Consistency	index	`yes/true` to activate, `no/false` to inhibit	Appears on Gateway requests when you enable the Gateway Configuration option **`EnhancedListingConsistency`**. (v9.3) Optionally supplied on a POST, PUT, COPY, APPEND, or DELETE request. Performs synchronous (search) indexing of the newly written/deleted object.

	sync	"now" or "wait" to activate, no/false to inhibit	Appears on Gateway requests when you enable the Gateway Configuration option **EnhancedListingConsistency**. (v9.3) Optionally used on a listing query GET request to force consistency of results that are returned on the listing. **"now"** per forms a refresh on the index in Elasticsearch immediately before preforming the listing query. **"wait"** delays execution of the listing query in order to give Elasticsearch time to refresh its index.
Metadata	preserve	yes/true to activate	Works with COPY, PUT, and APPEND requests to ensure that any custom metadata existing on the object is carried over on the write (see Custom Metadata Headers). To overwrite an existing value, include the header name with the new value on the request. Cannot be used with **replace**. (v9.2, v9.5)
	replace	yes/true to activate	Works with APPEND requests to remove any custom metadata existing on the object on the write, overriding the default APPEND behavior to preserve them (see Custom Metadata Headers). To add new metadata, include the header name with the new value on the request. Cannot be used with **preserve**. (v9.5)
Named Objects	newname	{new name for object, bucket, domain}	Provides a new name (within the same bucket only) for an update request (PUT, COPY, APPEND) on a named object. After you rename an object, requests for the original name return a 404 Not Found and the prior search metadata is removed. Remember that 'subdirectory' names are part of the object name, so they must be included as part of a **newname** query arg. The bucket name stops at the first slash, and everything after the first slash, including other slashes, are part of the object name. **newname** also lets you rename domains and buckets. See Renam ing Domains and Buckets.

	putcreate	`yes/true` to activate, `no/false` to inhibit	If set to yes, lets you use HTTP PUT Create to create new named objects. If the **`scsp.allowPutCreate`** storage setting is enabled, you do not need to add the **`putcreate`** query argument. If set to no, directs Swarm to treat the request as a regular PUT, generating a 404 Not Found error if the named object does not exist. Note Although domains and buckets are named, Swarm applies all PUT requests on these objects as updates, regardless of the setting. If `putcreate=yes` is used on a domain or bucket, Swarm fails the request with a 400 Bad Request error. See SCSP WRITE.
Multipart Write	inprogress	`yes/true` to activate	On a multipart PATCH complete, postpones HP segment consolidation until the object is completed again without the query argument, or by another method, such as COPY. (v9.4)
	shrink		On a multipart PATCH, required if the patch reduces the size of the object. (v9.4)
	partnumber	integer	On a multipart POST, indicates the part number of a multipart upload in progress. (v7.0)
	uploadid	upload id	On a multipart POST or DELETE, identifies all requests associated with a single multipart upload. It returns the ID as a 98-byte string. (v7.0)
	uploads	`yes/true` to activate	On a multipart POST, PUT, or APPEND, indicates that the request is a multipart upload write initiate. (v7.0)
Recursive Delete	recursive	`yes/true` to flag for deletion `now` for immediate reclamation	Required on DELETE of a bucket or domain (context). Indicates when the health process may begin asynchronously reclaiming any content contained in the deleted context. Unless you use **`recursive=now`**, this request creates a grace period based on the **`health.recursiveDeleteDelay`** Storage s etting (the default is one week). This grace period lets you recreate a bucket or domain before its content is lost. After the grace period ends, the health processor will begin deleting all of the content contained in the deleted domain or bucket. If the deleted domain or bucket contains undeletable objects, a critical error is logged that the object can be neither deleted nor accessed without its parent context. Added in v7.0.

Replicate on Write (ROW)	count	integer	Used to affect the `replicate=immediate` behavior.
	replicate	`immediate`, `full`, integer	Controls the response behavior to a POST, PUT, COPY, or APPEND request for a replicated object. • **`immediate`** – Only two replicas must be created before Swarm sends the response; if more replicas are required, those additional replicas are created after the response. • **`full`** – All of the required replicas must be created before Swarm sends the response. • **`integer`** – Specifies the number of replicas to be created. As with **`immediate`**, two replicas must be created before Swarm sends the response. However, if **`policy.replicas max`** or the lifepoint reps is less than this integer, then Swarm uses the smaller of those values. If the object in the request is a bucket or a domain, the above rules apply with **`scsp.maxContextReplicas`** taking the place of **`policy.replicas max`**.
Versioning	version	ETag of desired object version	Used on GET, HEAD, DELETE, COPY, APPEND, or SEND requests to specify a previous version to target on the request: `version={etag}` If used in contexts where versioning is disabled, operations that reference the current version proceed normally, but any other ETag results in a 404 - Not Found. (v9.2) If used on GET or HEAD requests of an immutable object (which cannot be versioned), the query will succeed if the query argument value is the UUID (and ETag) of the object. (v9.5)
	suspendversioning		Lets you temporarily suspend version creation on POST, PUT, COPY, APPEND, and DELETE requests for versioned objects. It has the effect of updating the current version without adding to the versioning chain. (v9.5)

Support and Administration Arguments

Name	Value(s)	Description and usage
admin	`yes/true` to activate	Use only with requests by the 'admin' user. An override used to rescue content from being stranded with no ability to delete or update it due to an overly restrictive Allow header.

aliasuuid	domain-UUID	Used only on an administrative override COPY request to specify the alias UUID (CID) of an existing domain or bucket for the purpose of changing its name. It must be used with the `newname` argument, which provides the new domain or bucket name. This is used to resolve name collisions that might occur during replication. See Resolving Duplicate Domain Names.
checkintegrity	`yes/true` to activate	Used on a GET of an erasure-coded object to get a summary of what segments exist in the manifest and their locations in the cluster. The body of the request has this information instead of the object. `checkintegrity` is a fast and efficient way to check the integrity of an erasure-coded object, for verifying that all segments exist before executing a GET. Swarm responds with 200 (OK) if it finds enough segments to recreate the object; if not, it returns 412 (Precondition Failed) and error text in the body of the response that lists the missing segments.
cid	UUID	A Context Identifier (CID) can be used to access an otherwise inaccessible object. See Accessing Inaccessible Objects with CID.
countreps	`yes/true` to activate or `diskless`	Used to have that HEAD return the number and location of replicas that are online in the cluster. For an erasure-coded object, the count returned is for the object manifest.
examine	`yes/true` to activate	On a GET or HEAD request, triggers an immediate health processor examination of the request object rather than waiting for the health processor to revisit the object as part of its normal cycle.
ignoreerrors	`yes/true` to activate	Used on an erasure-coded GET request to step over any EC sets that may be broken and, in effect, generate any data it can. Use of this query argument activates chunked transfer encoding.
recreatecid	domain or bucket alias UUID	Used only for certain administrative actions, such as a special POST request to recreate a particular domain or bucket with the specified alias UUID. Often, this is done after deleting a domain or bucket that orphans contents. See Restoring Domains and Buckets. Warning Do not attempt to use `recreatecid` as a way to move bucket contents across domains within the cluster.
redir	`yes/true` to activate, `no/false` to inhibit	Asks the request PAN to encourage or inhibit redirection on the request.

verbose	yes/true to activate	Used on a GET or HEAD request to have all relevant headers returned in the response. (v8.1) Includes a **MinReps** header that reports the number of replicas that should exist (applies to wholly replicated objects and segments).

SCSP Compatibility and Support

This section lists major API-level features and changes in Swarm releases starting with version 4.0 to assist you with writing client applications.

Determining the Swarm and SCSP proxy version

To determine which version of Swarm is running on a node, search for the Server header in any response by:

- Using a web browser.
- Sending a GET / request to the node using the node IP address or host name.
- Sending a GET / request to the node using the SCSP Proxy's external IP address or host name.

Using a browser

To search for the Server header using a web browser, use a browser with a head capture utility (such as Live HTTP Headers with Firefox) and enter the following URL in the Address field:

```
http://node-ip\[:scsp-port\]
```

where scsp-port is required only if you are using a value other than the 80 default value.

Using the node IP address or Host name

If your client application is on the same subnet as a Swarm node, send a GET / request to the node using the node's IP address or host name as the Host in the request.

In this example, the responding node is running Swarm version 5.1.

```
HTTP/1.1 200 OK
 Content-Type: text/html
 Content-Length: 733
 Cache-Control: no-cache
 Expires: Thu, 03 Jun 2011 19:09:05 GMT
 Age: 0
 Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
 Castor-System-TotalGBAvailable: 145
 Castor-System-TotalGBCapacity: 156
 Castor-System-Cluster: cluster.example.com
 Etag: "8c2c582c216a1f088c3652bced5a5f91"
 Date: Fri, 04 Mar 2011 22:55:45 GMT
 Server: CAStor Cluster/5.1.0
```

Using the SCSP proxy external IP address or Host name

If your client application is not on the same subnet as a Swarm node, send a GET / request to the Swarm SCSP Proxy, using the SCSP Proxy's external IP address or host name as the Host in the request.

In this example, the SCSP Proxy is running version 1.4 and Swarm is running version 6.0.

```
HTTP/1.1 200 OK
 Scsp-Proxy-Cluster: cluster.example.com
 Content-Type: text/plain
 Content-Length: 0
 Cache-Control: no-cache
 Expires: Tue, 13 Sep 2011 11:06:36 GMT
 Castor-System-TotalGBAvailable: 148567
 Castor-System-TotalGBCapacity: 349123
 Scsp-Proxy-Nodes: count=16
 SCSP-Proxy-Agent: SCSP Proxy Service/1.4.0
 Age: 0
 Etag: "6a04a4fef71925b92ec12de887ac4653"
 Via: 1.1 myhost.example.com (SCSP Proxy Service/1.4.0)
 X-Forwarded-For: myhost.example.com
 X-Forwarded-Server: myhost.example.com
 Date: Wed, 14 Sep 2011 14:53:16 GMT
 Server: CAStor Cluster/6.0.0
```

Issues with 100-Continue Header

Inconsistencies appear when the Swarm SDK is not integrated with the 100 Continue status header to implement the

SCSP protocol.

> Best practice for integrators
> Use the Swarm SDK, which includes full implementations of the SCSP protocol in multiple languages.

Integrators not using the SDK should be aware of the following inconsistencies with 100-Continue:

- Python httplib. The Python httplib wrapper does not alter its behavior in the presence of a 100-continue header and will send the complete request body without waiting for the continue response from the server. The Python SDK does not use httplib and does handle 100-continue headers correctly.
- C#/.NET WebClient/HttpWebRequest. HTTPWebRequest does not alter its behavior in the presence of a 100-continue header and will send the complete request body without waiting for the continue response from the server. Your client application is informed when it encounters a 100-continue header.
- Java Apache Commons HTTP client. The Apache commons HTTP client does handle 100-continue correctly after setting the POST method parameter:

```
method.getParams().setParameter(HttpMethodParams.USE_EXPECT_CONTINU
E, new Boolean(true) );
```

See RFC 7231 section 6.2.1 for more about the 100 Continue status.

SCSP Methods

These are general restrictions on methods:

- Immutables – The only supported methods for unnamed immutable objects are GET, HEAD, POST, and DELETE; other methods return a 403 (Forbidden) response.
- Domains – The only supported methods for domains are GET, HEAD, COPY, PUT, and APPEND. POST and DELETE require cluster administrator credentials and special consideration.

> Best practice
> If you cannot use the Content UI Overview to create, edit, and delete your domains, use the legacy Swarm Admin Console.

- SCSP APPEND
- SCSP COPY
- SCSP DELETE
- SCSP INFO
- SCSP READ
- SCSP SEND
- SCSP UPDATE
- SCSP WRITE

SCSP APPEND

This section provides general information about SCSP APPEND that applies to both named and unnamed objects.

APPEND is a request to the storage cluster to append arbitrary content data onto the end of an existing named object or aliased object while maintaining the previously populated metadata and object name or alias UUID. No whitespace or other characters are inserted by Swarm between the original and appended data. APPEND is not valid for immutable unnamed objects.

- Replicated — For replicated objects, the original content data is copied by Swarm from the original object and then the data supplied in the request is appended to it. Data appended to a replicated object can cause the object to become erasure-coded if the additional appended data pushes the object size over the configured policy.ecMinStreamSize threshold.
- Erasure-coded — APPENDs for previously erasure-coded objects with version 6.0 are optimized to write a new set of segments with the appended data and update the manifest, instead of rewriting the whole object to include the appended data as with replicated objects. If the EC constraints cannot be met on the APPEND request, the request will fail. For example, if encoding is 5:2 and there are only six nodes, the APPEND request will fail.

Special Query Arguments

replicate	Protects rapid updates	Important: Objects can be updated at a maximum frequency of once per second. Updating more frequently can cause unpredictable results with the stored object version. If your application updates objects faster than once per second, include the `replicate=immediate` query argument to ensure that more than one node can return the latest version in a subsequent read.
newname	Renames object	To rename a named object within the same bucket, use the newname query argument, which provides a new name with the update request (PUT, COPY, APPEND). After you rename an object, requests for the original name return a 404 Not Found and the prior search metadata is removed. (Note that the `newname` argument also lets you rename domains and buckets.)
preserve	Updates custom headers	APPEND only saves the existing headers, but the `preserve` argument updates the existing headers with those included on the request, if any. Cannot be used with `replace`. (v9.5)
replace	Replaces custom headers	APPEND only saves the existing headers, but the `replace` argument removes the existing headers and saves the new ones included on the request, if any. Cannot be used with `preserve`. (v9.5)

Guidelines for APPEND

- Include header for known or unknown size. You must include either the Content-length or Transfer-Encoding: chunked header.
  - If you know the size of the object, use the Content-length header. The Content-length value must specify

  the correct length of the appended content data. The Content-length header in the object is adjusted to reflect the actual length of the original data plus the appended data.
  - If you do not know the size of the object (such as a live video feed), use the Transfer-Encoding: chunked header (or the UNDETERMINED_LENGTH parameter if using the SDK). This header tells Swarm that the size of the appended data is unknown. Do not combine this header with the Content-length header. All other headers stored with the object are copied without change to the newly-updated object. As a result, the x-acme-meta-* and lifepoint headers in the preceding examples are ignored.

- Content-MD5 Headers corrected. If you provide a Content-MD5 header with the APPEND request, Swarm computes the digest of the content data plus the appended data and compares it with the provided MD5 hash. This assumes you either have access to the original data or maintain a running digest to which appended data is added before each APPEND request. If a Content-MD5 header was persisted with the original object, it is removed when new data is appended to the object. Any new, correct Content-MD5 supplied with an append is persisted with the new revision and returned on any subsequent GET or HEAD.

- Omit Range Headers. Range headers are incompatible with the APPEND method. Including a Range header with an APPEND request results in a 400 Bad Request error response. Other aspects of the APPEND method, including response codes, are the same as PUT.

APPEND for named and alias objects

The syntax of an APPEND request is similar to a PUT. As with PUT, the object name or UUID returned after a successful APPEND matches the one supplied in the request. APPEND for an alias object is the same as for a named object except that you use the object's UUID instead of a name on the first line of the command.

Example APPEND for named object

```
APPEND /mybucket/samplefile.txt HTTP/1.1
  Host: cluster.example.com
  Content-length: 29
  x-acme-meta-color: blue
  x-acme-meta-weight: 42
  lifepoint: [Sunday, 06-Nov-2010 08:49:37 GMT] reps=3, deletable=no
  lifepoint: [] delete
  Status: Approved
```

APPEND for unnamed objects

APPEND is not supported for unnamed immutable objects: the UUID must be an alias object. The query argument alias =yes is optional as of v7.0.

Swarm returns a 404 Not Found error when the object is not alias (appending to an immutable object or to a UUID that does not exist).

Normal responses to APPEND

The APPEND method returns a response only after all of the data is copied and the update is complete.

> For a list of response headers, see SCSP Headers.

Error responses to APPEND

If you execute the APPEND method on an object in a domain but the domain does not exist or is not in the content cache on the node that receives the request, Swarm responds with 409 Conflict.

SCSP COPY

This section provides general information about SCSP COPY that applies to both named and alias objects. Immutable unnamed objects cannot have their metadata updated.

- Best practice
- Headers to preserve
- COPY for named objects
- COPY for alias objects
- Normal responses to COPY
- Error responses to COPY

The COPY method lets you update the metadata on an object after the initial write. COPY lets you change headers without changing content, such as to update permissions or change a lifepoint. That is, the COPY method does not cre ate any new objects: rather, it updates the metadata on an existing object, by copying its content verbatim while replacing its metadata.

Best practice

COPY modifies all headers at once, so simply calling COPY with your new header or modified header value has the effect of dropping all other user-supplied headers that were originally set on the object.

To use COPY correctly, follow this process:

1. HEAD the original object.
2. Grab all writable persisted headers (see "Headers to preserve", below).
3. COPY the object with those header values, with these changes:
   a. Add any new headers or updated values.
   b. Omit any headers that you want to remove.

The requestor decides exactly which headers will be written. The COPY method returns a response only after the object update is complete.

Update frequency

Named and alias objects can be updated at a maximum frequency of once per second. Updating more frequently can cause unpredictable results with the stored object version. If your application updates objects faster than once per second, include the replicate query argument to ensure that more than one node can return the latest version in a subsequent read.

Gateway transforms

In the Gateway, domain admins can specify header transformations for POSTs, PUTs, and COPYs. A COPY sent through the Gateway is subject to the transform rules, replacing all headers that match the applicable transform rule header names with values in the rules. This means that Gateway will discard any headers in a

COPY request that match transform rules and will update those headers with current request values for rule Substitution Variables. That is, COPY will replace date and user variables with the current request values rather than the original values on the object.

Headers to preserve

Following are persisted headers that you will typically want to preserve when you add metadata to an object:

- Allow
- Allow-Encoding
- Cache-Control
- Castor-* (except those with System)
- Content-Base
- Content-Disposition
- Content-Encoding
- Content-Language
- Content-Length
- Content-Location
- Content-MD5
- Content-Type
- Expires
- Lifepoint
- Policy-* (except those with Evaluated[-Constrained])
- X-*-Meta[-*]

Tip
Add the preserve query argument to the COPY request to ensure that any custom metadata existing on the object is carried over to the copy. To overwrite an existing value, include the header name with the new value on the request. (v9.2)

> See SCSP Headers for details about using these headers with COPY.

COPY for named objects

The syntax of a COPY request is similar to an empty PUT request on an alias object.

```
COPY /some/filename HTTP/1.1
  Host: cluster.example.com
  Content-Length: 0
  x-xml-meta-data-color: blue
  x-xml-meta-data-weight: 42
  x-xml-meta-data:
<size>large</size><color>blue</color><specialorder/>
  lifepoint: [Sun, 06 Nov 2010 08:49:37 GMT] reps=3, deletable=no

  lifepoint: [] delete
```

Renaming – To rename a named object within the same bucket, use the newname query argument, which provides a new name with the update request (PUT, COPY, APPEND). After you rename an object, requests for the original name return a 404 Not Found and the prior search metadata is removed. (Note that the newname argument also lets you rename domains and buckets.)

Behavior of COPY

- Rewrites EC manifest. If an erasure-coded object is modified by COPY, COPY rewrites the object manifest instead creating a new object with new metadata. (See Working with Large Objects.)
- Has no content body. The Content-Length header is optional. If you include this header, its value must be 0 because there is no content body for a COPY request.
- Replaces metadata headers. Additional headers (such as the two x-*-meta-* headers in the example and the lifepoint headers), replace all existing metadata in the original object. The only exception is the Content-Length header value that continues to provide the original length of the content data.
- Calculates and compares hashes.
  - Non-EC object - If the client provides a Content-MD5 header with the COPY request, Swarm recomputes the digest of the content data as it copies it and compares it with the provided MD5 hash. Similar to a WRITE, if the provided and calculated hashes do not match, the operation will fail.
  - EC object - The request calculates a new MD5 on non-EC objects only. On an EC object, the `gencontentmd5` query argument (or the deprecated `Expect: Content-MD5` header) or `content-md5` header is allowed only if the existing object already has a content-md5 stored on it. If so, any new value must match the existing value.
- Responses. Other aspects of the COPY method, including response codes, are the same as PUT. The COPY method returns a response only after all the data has been copied and the object update is complete.

COPY for alias objects

The UUID returned after a successful COPY is identical to the UUID supplied in the request, which is similar to a PUT or APPEND request.

The query argument alias=true is an optional acknowledgment that the method will be executed on an existing alias object. The object specified by the included UUID must be an alias object in Swarm.

Failure to perform a COPY on an alias object results in a 403 (Forbidden) response.

Normal responses to COPY

A multipart COPY by part behaves like a multipart write completion, sending back a 202 response code and keep-alive characters to prevent client timeouts. (v9.1.2)

> For a list of response headers, see SCSP Headers

Error responses to COPY

If you execute the COPY method on an object in a domain but the domain does not exist or is not in the content cache on the node on which the request is directed, Swarm responds with 409 (Conflict).

SCSP DELETE

This section provides general information about SCSP DELETE that applies to both named and unnamed objects.

DELETE is a request to the storage cluster to remove a specific object. The DELETE request is formatted as a simple HTTP request using the DELETE method.

SCSP Method	HTTP Method	RFC 7231 Section
DELETE	DELETE	4.3.5

DELETE for named objects

```
DELETE /bucket/photo.jpg HTTP/1.1
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
```

Guidelines for DELETE

- For bucket requests, use a separate initialization or setup routine that runs less frequently. Swarm is optimized for calls on individual objects, not domains or buckets (which are centralized resources), so do not make bucket calls on the high-availability code path of your client application.
- Reuse object names. After a named object is deleted, another object with the same name can be created in the same bucket. Unlike unnamed objects, whose UUIDs are not reused, names can be reused.
- Pause before recreating. Deleting a named object involves an underlying update, for Swarm to write a special marker value to the name. When recreating a named object after deleting it, be sure to wait at least one second.

DELETE for domains and buckets

When you delete a domain or bucket, you need to also delete any objects contained within it, or else these objects will be orphaned, lost, and consuming disk space unnecessarily. These deletes are recursive, iterating until every object contained in the domain or bucket is dealt with.

> Note
>
> If you try to delete a domain or bucket without having a recursive argument or parameter in force, Swarm generates an SCSP error.

To delete a domain or bucket, you must include the recursive query argument:

- recursive | recursive=true|yes
  - Grants a 1-week grace period (default) during which you can restore the domain or bucket before the health processor begins reclaiming the space.
  - To change the length of the grace period, edit the health.recursiveDeleteDelay parameter.
- recursive=now
  - Grants no grace period. The health processor begins reclaiming the space immediately.

If you have existing integrations that do not use the recursive argument, you can avoid changing them by adding a global configuration parameter: scsp.autoRecursiveDelete=True.

> If you erroneously deleted a domain or bucket, you can get it back without data loss if it is within the grace period. See Restoring Domains and Buckets.

Reusing bucket names

You can delete a bucket and recreate another bucket with the same name. However, be aware of the following:

- The new bucket is a different bucket that happens to have the same name.
- After you delete a bucket, all objects in that bucket are inaccessible, even if you subsequently create another bucket with the same name.

> Best practice
>
> Wait at least twice the value of cache.realmStaleTimeout before you attempt to recreate a bucket with the same name as a bucket just deleted: the default is 600 seconds (10 minutes), so you should wait 20 minutes, then create the new bucket. This waiting period applies only to reusing names of buckets: deleting a named object and recreating an object with that name requires only a 1-second pause.

DELETE for unnamed objects

```
DELETE /7A25E6067904EAC8002498CF1AE33023 HTTP/1.1
  User-Agent: Swarm Client/0.1
```

> Note

Regardless of the value of the enforceTenancy setting, no domain specification is needed or recognized for GET, HEAD, or DELETE requests on unnamed (alias or immutable) objects.

If the method succeeds, the content associated with the name or UUID supplied in the request is no longer available. This does not imply that all copies of the content were erased. The cluster now responds to any READ request for that UUID with a 404 Not Found error.

Note

Swarm deletes both the manifests and the segments of an erasure-coded object. Erasure coding lets you store larger objects with a smaller footprint. See Working with Large Objects.

An object can be deleted by an application or lifecycle as follows:

- Application deleting an object - All online replicas in a single cluster are removed immediately after a delete method is executed on an object. (An online replica is one that resides on a cluster node that is on line at the time the delete is issued.) In addition, the cluster remembers the name or UUID has been deleted for 14 days, in the event that one or more nodes holding replicas of the deleted object are off line at the time the delete was issued.
- Lifecycle deleting an object - An object can have a storage policy defined by the application and stored along with it. Part of the storage policy might be an expiration period, beyond which the object is to be removed. In the case of a policy-defined deletion, all replicas, wherever they are stored, are deleted at approximately the same time and become unavailable at most one second after the expiration date and time.

Note

The UUID of a deleted object is never reused, even if the object is mutable.

DELETE for alias objects

To delete alias objects, add a query argument alias:

```
DELETE /7A25E6067904EAC8002498CF1AE33023?alias HTTP/1.1
  User-Agent: Swarm Client/0.1
```

Note

Regardless of the value of the enforceTenancy setting, no domain specification is needed or recognized for GET, HEAD, or DELETE requests on unnamed (alias or immutable) objects.

Normal Responses to DELETE

- DELETE response
- DELETE response for moved permanently
- DELETE response for moved temporarily

> For a list of response headers, see SCSP Headers.

DELETE response

The following response indicates that the content was found and is being deleted:

```
HTTP/1.1 200 OK
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2010 18:16:54 GMT
  Content-Type: application/x-www-form-urlencoded
  Last-Modified: Fri, 15 Oct 2010 18:16:54 GMT
  Etag: "06eec5e2c3f1aadcb41ef7fd52adc049"
  Content-Length: 0
  Date: Fri, 15 Oct 2010 21:43:51 GMT
  Server: CAStor Cluster/5.0.0
```

DELETE response for moved permanently

The following response indicates the content can be deleted as requested, but another node in the cluster will complete the Delete. Additionally, all future requests of this storage cluster should be made through the new access node until another 301 response is received. The value of the Location header indicates which node in the cluster should receive the redirect. The client is expected to send another DELETE request using the exact URI contained in the Location header.

```
HTTP/1.1 301 Moved Permanently
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Location:
http://node-ip/name-or-uuid?auth=2096EFA659295BBB819D1FECCE77D2EF
  Content-Length: 0
```

DELETE response for moved temporarily

The following response indicates the content can be deleted as requested, but another node in the cluster will complete the Delete. All future requests of this storage cluster should be made through the new access node until another 301 response is received. The value of the Location header indicates which node in the cluster is assigned the redirect. The client is expected to send another DELETE request using the exact URI contained in the Location header.

```
HTTP/1.1 307 Temporary Redirect
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster 5.0.0
  Location:
http://node-ip/name-or-uuid?auth=2096EFA659295BBB819D1FECCE77D2EF
  Content-Length: 0
```

This response is similar to the 301 response, except the client should continue to use the current node (the one generating this response) for future requests until further notice.

Error Responses to DELETE

The storage cluster could return the following responses when the specified content cannot be deleted from the cluster or there is a problem with the DELETE request.

400 Bad Request	The following response indicates a problem with the DELETE request, such as missing mandatory headers, invalid message body, or any other violation of HTTP/1.1 by the DELETE request. The reason for the error is included in the message body of the response.
	`HTTP/1.1 400 Bad Request`
	`  Date: Wed, 1 Sept 2010 15:59:02 GMT`
	`  Server: CAStor Cluster/5.0.0`
	`  Connection: close`
	`  Content-Type: text/html`
	`  Content-Length: 24`
	`  CRLF Host header is required.`

404 Not Found	The following response indicates the requested object could not be located in this cluster.
	`HTTP/1.1 404 Not Found`
	`  Date: Wed, 1 Sept 2010 15:59:02 GMT`
	`  Server: CAStor Cluster/5.0.0`
	`  Content-Length: 56`
	`  Content-Type: text/html`
	`  CRLF The requested resource is not available in this`
	`cluster.`

<table>
<tr><td>403
Forbidden</td><td>The following response indicates the requested object could not be deleted because its current state forbids it. An object's lifecycle might include periods of time when users are not allowed to delete it from the cluster.<pre>
HTTP/1.1 403 Forbidden
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster/5.0.0 `
  Content-Length: 56
  Content-Type: text/html
  CRLF
  The requested resource cannot be deleted at this time.
</pre></td></tr>
</table>

SCSP INFO

This section provides general information about SCSP INFO that applies to both named and unnamed objects.

- INFO for named objects
- INFO for unnamed objects
- INFO for alias objects
- Normal responses to INFO
- Error responses to INFO

INFO is a request to the storage cluster to provide information about a specific object. The INFO message is identical in semantics to the READ request, except that the object (if found) is not returned in the response. If the referenced content is found in the cluster, only the meta-information about that object is returned in the form of response headers. The INFO request is formatted as a simple HTTP request using the HEAD method.

SCSP Method	HTTP Method	RFC 7231 Section
INFO	HEAD	4.3.2

> Note
> SCSP allows HEAD requests with mismatched `Accept-Encoding` headers to receive responses as if the encoding matched that of the object. Swarm relaxed this RFC restriction because a HEAD request returns no body contents, so there is nothing to encode.

INFO for named objects

```
HEAD /mybucket/samplefile.txt HTTP/1.1
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
```

INFO for unnamed objects

```
HEAD /06eec5e2c3f1aadcb41ef7fd52adc049 HTTP/1.1
  User-Agent: Swarm Client/0.1
```

Note

Regardless of the value of the enforceTenancy setting, no domain specification is needed or recognized for GET, HEAD, or DELETE requests on unnamed (alias or immutable) objects.

INFO for alias objects

To INFO an alias object, optionally add the alias=yes query argument to the URI portion of the HTTP request line, as shown below.

```
HEAD /41A140B5271DC8D22FF8D027176A0821?alias=yes HTTP/1.1
  User-Agent: Swarm Client/0.1
```

Note

Regardless of the value of the enforceTenancy setting, no domain specification is needed or recognized for GET, HEAD, or DELETE requests on unnamed (alias or immutable) objects.

Normal responses to INFO

> For a list of response headers, see SCSP Headers.

The responses described in this section might be returned by the storage cluster in the case where the requested content was found. The content's meta-information can be returned directly by the node that received the request or the request might be redirected to another node in the cluster.

Note

> If you add `?checkIntegrity` to HEAD and GET requests for the same object, you would see different Content-Length values in the responses. This occurs because the HEAD response returns the Content-Length of the manifest rather than the object.

The following response indicates that the requested object was located, but that another node in the cluster will service the request for meta-information. Additionally, all future requests of this storage cluster should be made through the new access node until another 301 response is received. There is no message-body, so the content length is always 0. The value of the Location header indicates which node in the cluster should receive the redirect.

301 Moved Permanently

The client is expected to send another INFO request using the exact URI contained in the Location header.

```
HTTP/1.1 301 Moved Permanently
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster/v8b2
  Connection: close
  Location:
http://node-ip/name-or-uuid?auth=2096EFA659295BBB819D1FECCE77D2EF
  Content-Length: 0
```

307 Temporary Redirect

The following response (307) is similar to the 301 response, except the client should continue to use the current node (the one generating this response) for future requests until further notice.

```
HTTP/1.1 307 Temporary Redirect
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Location:
http://node-ip/name-or-uuid?auth=2096EFA659295BBB819D1FECCE77D2EF
  Content-Length: 0
```

Normal responses for named objects

```
INFO response for a named object in a domain:
HTTP/1.1 401 Unauthorized
  WWW-Authenticate: Digest realm="cluster.example.com/_administrators",
  nonce="05d0a60ee1f44361f449496505e05116",
opaque="fd9c8e14e20fb7c13408c049b7d222af",     `
  stale=false,   `
  qop="auth",   `
  algorithm=MD5
  WWW-Authenticate: Basic realm="cluster.example.com/_administrators"
  Content-Length: 51
  Content-Type: text/html
  Date: Sat, 16 Oct 2010 00:23:24 GMT
  Server: CAStor Cluster 5.0.0
  Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
HTTP/1.1 200 OK
  Castor-System-Alias: ec87e3c7c410cc04fc4c838061898d9c
  Castor-System-CID: ffffffffffffffffffffffffffffffff
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2010 23:59:40 GMT
  Castor-System-Name: cluster.example.com
  Castor-System-Owner: admin@CAStor administrator
  Castor-System-Version: 1287187180.959
  Content-Length: 0
  Last-Modified: Fri, 15 Oct 2010 18:35:56
  GMT lifepoint: [] reps=16
  Etag: "da8bfbb04d089b9c22ae77747f327233"
  Date: Sat, 16 Oct 2010 00:23:24 GMT
  Server: CAStor Cluster/5.0.0
```

The initial 401 Unauthorized response is a normal initial response to HTTP authentication. Because access to a domain always requires administrator credentials, you will always see an initial 401 on an INFO on a domain.

```
HTTP/1.1 200 OK
  Castor-System-Alias: d36dfca69ba7752f4708b1fa9bf9918b
  Castor-System-CID: ec87e3c7c410cc04fc4c838061898d9c
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2010 18:36:05 GMT
  Castor-System-Name: bucket
  Castor-System-Version: 1287167765.255
  Content-Length: 0
  Content-Type: application/x-www-form-urlencoded
  Last-Modified: Fri, 15 Oct 2010 18:36:05 GMT
  Etag: "21641b39f4fdc1e86dc67e798a320980"
  Date: Fri, 15 Oct 2010 23:54:44 GMT
  Server: CAStor Cluster/5.0.0
HTTP/1.1 200 OK
  Castor-System-CID: d36dfca69ba7752f4708b1fa9bf9918b
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2010 22:09:19 GMT
  Castor-System-Name: file.txt
  Castor-System-Version: 1287180559.436
  Content-Length: 26
  Content-Type: application/x-www-form-urlencoded
  Last-Modified: Fri, 15 Oct 2010 22:09:19 GMT
  Etag: "c744aa90d375aa3e1f228f74b7960e54"
  Date: Fri, 15 Oct 2010 23:51:44 GMT
  Server: CAStor Cluster/5.0.0
```

The response for a named object is very similar to the response for a bucket except that Castor-System-CID is the identifier of the named object's parent (the bucket).

Normal responses for unnamed objects

The following response indicates that the node that received the request found the requested content and is returning meta-information about the object in the headers of this response.

```
HTTP/1.1 200 OK
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Content-Type: image/jpeg
  Content-Length: (length of the content of the Swarm object)
  Replica-Count: (number of replicas in cluster)
  [ application-meta-information ]
```

Error responses to INFO

The responses in this section may be returned by the storage cluster when the specified content cannot be found or there is a problem with the INFO request.

The following response indicates a problem with the INFO request, such as missing mandatory headers, invalid message body, or any other violation of HTTP/1.1 by the HEAD request. The reason for the error is included in the status line.

```
HTTP/1.1 400 Bad Request
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster 5.0.0
  Content-Length: 24
  Content-Type: text/html
```

Indicates that the requested object could not be located in this cluster:

```
HTTP/1.1 404 Not Found
  Date: Wed, 1 Sept 2010 15:59:02 GMT
  Server: CAStor Cluster 5.0.0
  Content-Length: 56
  Content-Type: text/html
```

- Getting Replica Counts and Location
- Getting Node Status and Cluster Capacity

Getting Replica Counts and Location

Use the query argument countreps=yes to request INFO to return the number and location of object replicas that are online in the cluster. (This was previously the default behavior but was changed as of 4.0 so that replicas are not counted unless specifically requested.)

To enable Swarm to return the replica count and location of an object, include the countreps=yes query argument:

```
HEAD /mybucket/samplefile.txt?countreps=yes HTTP/1.1
 Host: cluster.example.com
 User-Agent: Swarm Client/0.1
```

When completed, Swarm returns the replica count and location.

```
HEAD / HTTP/1.1 HTTP/1.1 200 OK
 Castor-System-Cluster: cluster.example.com
 Castor-System-Created: Sun, 07 Jul 2013 17:10:06 GMT
 Content-Length: 4
 Content-Type: application/x-www-form-urlencoded
 Last-Modified: Sun, 07 Jul 2013 17:10:06 GMT
 Location: http://192.168.1.6:80/70ef3152c831c2c80bbce6505dfb7d0a
 Volume: 992c9259b37637927cec444bf9865b8c
 Location: http://192.168.1.52:80/70ef3152c831c2c80bbce6505dfb7d0a
 Volume: 4d2ffe4f5b8403af3bb9b5408c1babf7
 Replica-Count: 2
 Etag: "70ef3152c831c2c80bbce6505dfb7d0a"
 Volume: 992c9259b37637927cec444bf9865b8c
 Volume-Hint: 4d2ffe4f5b8403af3bb9b5408c1babf7
 Entity-MD5: 5r0hE+hjVdcj6owxoDRhaw==
 Stored-Digest: e6bd2113e86355d723ea8c31a034616b
 MinReps: 2
 Date: Sun, 07 Jul 2013 17:10:33 GMT
 Server: CAStor Cluster/6.1.0
```

Getting Node Status and Cluster Capacity

An INFO request submitted without a domain, a name, or a UUID returns basic status information:

- Counts for high-level node methods (such as READ, WRITE, DELETE)
- Cluster-wide values for Space Available and Total Capacity.

```
HEAD / HTTP/1.1
 Host: cluster.example.com
 User-Agent: Swarm Client/0.1
```

Tip
This information is also available in the Swarm Storage UI

SCSP READ

- READ for named objects
- READ for unnamed objects
- READ for erasure-coded objects

- READ with content validation
- READ for node status and cluster capacity
- READ with range headers

This section provides general information about the SCSP READ method that applies to both named and unnamed objects.

READ is a request to the storage cluster for a specific object. The READ request is formatted as a simple HTTP request using the GET method.

SCSP Method	HTTP Method	RFC 7231 Section
READ	GET	4.3.1

READ for named objects

```
GET /mybucket/samplefile.txt HTTP/1.1
Host: cluster.example.com
User-Agent: Swarm Client/0.1
```

READ for unnamed objects

```
GET /12BFEA648C2697A56FD5618CAE15D5CA HTTP/1.1
Host: cluster.example.com
User-Agent: Swarm Client/0.1
```

Swarm makes no assumptions about User-Agent (except that it is an HTTP/1.1 client). The Host header is mandatory and must conform to the requirements of Section 14.23 in the HTTP/1.1 specification.

Note

Regardless of the value of the enforceTenancy setting, no domain specification is needed or recognized for GET, HEAD, or DELETE requests on unnamed (alias or immutable) objects.

READ for erasure-coded objects

Erasure coding objects on disk lets you store large objects in the cluster with a smaller storage footprint, compared to earlier versions of Swarm.

> See Working with Large Objects.

READ is affected by the checkIntegrity=yes query argument that is used to verify that all segments are found before executing the READ.

> See SCSP Query Arguments.

READ with content validation

To validate the content during a read, add the query argument validate=yes to the URI:

```
GET /name-or-uuid?validate=yes HTTP/1.1
Host: cluster.example.com
User-Agent: Swarm Client/0.1
```

Using this argument, the content digest is computed and compared at the end. A hash mismatch causes the socket connection to be dropped before sending the final bytes.

READ for node status and cluster capacity

A READ request can be submitted without a name or UUID:

```
GET /  HTTP/1.1
Host: cluster.example.com
User-Agent: CAStor Client/0.1
```

Using this argument returns counts for high-level node methods (READ, WRITE, DELETE, …), as well as cluster-wide Space Available and Total Capacity values.

This information is also available on the Node Status page that appears when you navigate to a node's IP address with the designated SCSP port (for example, `http://192.168.99.100:80`).

READ with range headers

In some cases, an application might be interested in only a byte portion of a larger object stored in Swarm. Rather than read the entire object and filter out the interesting parts, the application can include one or more Range headers with an SCSP READ request. A READ request can include more than one Range header.

> See the HTTP/1.1 specification for a thorough discussion of Range headers.

Below are some examples and their interpretations:

Range	Returns
0-499	First 500 bytes of the object
500-999	The second 500 bytes
-500	Last 500 bytes
0-499, 500-999	First 1000 bytes

> Note
> Range headers are not compatible with either integrity seals or the ContentMD5 header because both require a hash of the object's entire contents. If the Range is not set to a value greater than or equal to the size of the object, the connection is closed as if the integrity seal or the Content-MD5 was invalid.

READ requests that include invalid Range headers (for example, ranges that do not exist in the object) cause Swarm to respond with a 416 (Range not satisfiable) error. A successful response to a READ that includes one or more Range headers is 206 (Partial content). Of course, only data in the requested ranges are included in the 206 response.

READ requests that include a range such as Range: 0-199, 300-349, 500-999 returns a Content-Type: multipart/byteranges response consisting of three parts: 200, 50, and 500 bytes of content.

> See 14.35 (Range) and appendix 19.2 (Internet Media Type multipart/byteranges) in HTTP/1.1 RFC .

> Note
> Entering a range with the range in reverse order (where the end of the range is entered first) will return the entire object. For example, Range: 999-500 returns all of the content in the object. The range header is essentially ignored.

- Normal Responses to READ
- Error Responses to READ

Normal Responses to READ

The storage cluster can return the following responses for the requested content:

- READ response for domain
- READ response for bucket
- READ response for named object
- READ response for unnamed objects
- READ response for range headers
- READ response for moved permanently
- READ response for moved temporarily

The content can be returned directly to the node that sent the request or redirected to another node in the cluster.

> For a list of response headers, see SCSP Headers.

READ response for domain

Example response to a READ or INFO request for an object in a domain.

The initial 401 Unauthorized response is a normal part of HTTP authentication for a domain. Because access to a domain requires domain manager credentials, you will always see an initial 401 Unauthorized response for an INFO request in a domain.

```
HTTP/1.1 401 Unauthorized
  WWW-Authenticate: Digest
  realm="cluster.example.com/_administrators",
  nonce="05d0a60ee1f44361f449496505e05116",
  opaque="784d8bc3fe3a48a5105b4f8ddd8ae0e7",
  stale=false, qop="auth", algorithm=MD5
  WWW-Authenticate: Basic
  realm="cluster.example.com/_administrators"
  Content-Length: 51
  Content-Type: text/html
  Date: Sat, 16 Oct 2012 00:41:23 GMT
  Server: CAStor Cluster 5.0.0
  Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
HTTP/1.1 200 OK
  Castor-Authorization: cluster.example.com/_administrators, POST=
  Castor-System-Alias: ec87e3c7c410cc04fc4c838061898d9c
  Castor-System-CID: ffffffffffffffffffffffffffffffff
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2012 23:59:40 GMT
  Castor-System-Name: cluster.example.com
  Castor-System-Owner: admin@CAStor administrator
  Castor-System-Version: 1287187180.959
  Content-Length: 0
  Last-Modified: Fri, 15 Oct 2012 18:35:56
  GMT lifepoint: [] reps=16
  Etag: "da8bfbb04d089b9c22ae77747f327233"
  Date: Sat, 16 Oct 2012 00:41:23 GMT
  Server: CAStor Cluster/5.0.0
```

READ response for bucket

Example response to a READ request for a bucket:

```
HTTP/1.1 200 OK
  Castor-System-Alias: d36dfca69ba7752f4708b1fa9bf9918b
  Castor-System-CID: ec87e3c7c410cc04fc4c838061898d9c
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2012 18:36:05 GMT
  Castor-System-Name: bucket
  Age: 62
  Castor-System-Version: 1287167765.255
  Content-Length: 0
  Content-Type: application/x-www-form-urlencoded
  Last-Modified: Fri, 15 Oct 2012 18:36:05 GMT
  Etag: "21641b39f4fdc1e86dc67e798a320980"
  Date: Fri, 15 Oct 2012 19:00:46 GMT
  Server: CAStor Cluster 5.0.0
```

READ response for named object

Example response to a READ request for a named object:

```
HTTP/1.1 200 OK
  Castor-System-CID: d36dfca69ba7752f4708b1fa9bf9918b
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2012 18:37:08 GMT
  Castor-System-Name: file.txt
  Castor-System-Version: 1287167828.514
  Content-Length: 11
  Content-Type: application/x-www-form-urlencoded
  Last-Modified: Fri, 15 Oct 2012 18:37:08 GMT
  Etag: "a896b8e88fe7fc15c9b8f9b2d19e311d"
  Date: Fri, 15 Oct 2012 18:45:12 GMT
  Server: CAStor Cluster/5.0.0
```

READ response for unnamed objects

Example response to a READ request for an unnamed object.

```
HTTP/1.1 200 OK
  Castor-System-Cluster: cluster.example.com
  Castor-System-Created: Fri, 15 Oct 2012 18:16:54 GMT
  Content-Type: text/html
  Content-Length: 645
  Cache-Control: no-cache
  Expires: Tue, 05 Oct 2012 19:40:23 GMT
  Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
  [ application meta-information ]
  Date: Wed, 06 Oct 2012 23:27:03 GMT
  Server: CAStor Cluster/5.0.0
  [ content ]
```

This response means the node receiving the request is returning the requested content in the message body of the response.

READ response for range headers

When a READ request includes one or more range headers, a successful Swarm response includes only the requested bytes range(s). In this case, instead of returning a 200 OK response, Swarm returns a 206 Partial Content response, indicating that only part of the content is being returned. The application-meta-information and content-stream are the same.

```
HTTP/1.1 206 Partial Content
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster 5.0.0
  Content-Length: 500
  [ application-meta-information ]
  CRLF
  [ content ]
```

READ response for moved permanently

The following response shows that the requested object was located, but another node in the cluster will supply the content. Additionally, all future requests of this storage cluster should be made through the new access node until another 301 response is received.

There is no message-body, so the content length is always 0. The value of the Location header indicates which node in the cluster receives the redirect.

The client is expected to send another READ request using the exact URI in the Location header.

```
HTTP/1.1 301 Moved Permanently
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Location:
http://cluster-ip/name-or-uid?auth=2096EFA659295BBB819D1FECCE77D2EF
  Content-Length: 0
```

READ response for moved temporarily

The following response is similar to the 301 response, except the client should continue to use the current node (the one generating this response) for future requests until further notice.

```
HTTP/1.1 307 Temporary Redirect
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Connection: close
  Location:
http://cluster-ip/name-or-uid?auth=2096EFA659295BBB819D1FECCE77D2EF
  Content-Length: 0
```

Error Responses to READ

The storage cluster can return various types of responses when the specified content cannot be found or there is a problem with the READ request.

- 400 Bad Request for READ
- 404 Not Found for READ
- 416 Requested range not satisfiable for READ

400 Bad Request for READ

The response below indicates a problem with the READ request, such as missing mandatory headers, invalid message body, or any other violation of HTTP/1.1 by the GET request. The actual reason for the error is described in the message body of the response.

```
HTTP/1.1 400 Bad Request
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Connection: close
  Content-Length: 24
  Content-Type: text/html
  CRLF Host header is required.
```

404 Not Found for READ

The following response indicates that the requested content could not be located in this cluster because of one of the following reasons:

- The object was deleted.
- The request included errors (for example, the wrong requested bucket name).
- Network failure.
- The node (all nodes that contain the requested object) is down for maintenance.
- Timeouts occur due to a heavily loaded or extremely active cluster.

This response can occur after you add additional drive capacity to the cluster. When the cluster recognizes the new storage drive, it tries to rebalance the objects from heavily loaded nodes to less-loaded nodes.

Applications that recognize a given object exists should retry the request after a 404 error message.

```
HTTP/1.1 404 Not Found
  Content-Length: 97
  Content-Type: text/html
  Date: Tue, 14 Jun 2012 01:12:16 GMT
  Server: CAStor Cluster/6.0.0
  Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
  <html><body><h2>CAStor Error</h2><br>Bucket example.com/bucket failed
to load (404)</body></html>
```

416 Requested range not satisfiable for READ

The response below indicates that one or more range headers supplied in the request were out of bounds with respect to the data.

```
HTTP/1.1 416 Requested range not satisfiable
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Connection: close
  Content-Length: 24
  Content-Type: text/html
  Range specs out of range
```

SCSP SEND

The SCSP SEND method applies to both named and unnamed objects. The SEND request lets you explicitly transmit a single object from a source cluster to a remote one, such as for keeping two clusters immediately synchronized. This feature is best used with a replication feed, which will act as a catch-up mechanism if the intracluster network is down or the SEND command should fail.

SEND Requests

SEND can only be used by an `admin` user. WIth a SEND request, Swarm performs the appropriate GET/retrieve in the destination cluster to verify whether the object already exists there.

> Note
>
> You issue SEND directly on a Swarm node in the source cluster, to transfer a single object to a destination cluster. Do not send the request to a Gateway or other proxy.

You use the following headers with SEND requests:

Castor-System-Cluster	required	The value of the `cluster.name` setting of the destination cluster.
Castor-System-Target	required	The `IP:port` of a node or reverse proxy of the destination cluster.
Castor-System-Auth	optional	The `username:password` for an administrative account on the remote cluster, only if it differs from the source cluster.

Responses to SEND

The request returns information about that request in the body of the response. SEND behaves much like a HEAD request, with the headers of the response resembling that of a HEAD request.

> Important
>
> The SEND response is chunked transfer encoded, so the client of the SEND request must be prepared for chunked transfer encoding. The response body may contain additional leading newlines sent incrementally, which keeps the connection open in long requests.

The most common response codes are these:

- 201 - the object has been transferred successfully
- 409 - the object already exists in the destination cluster

Example of SEND

The following request transfers the object to the "dr" cluster at 192.168.1.13, with a 201 in that cluster:

```
curl -i -X SEND --location-trusted
 -H "Castor-System-Cluster: dr"
 -H "Castor-System-Target: 192.168.1.13:80"
 --anyauth -u admin:ourpwdofchoicehere
 "http://192.168.1.12:80/97f7149dec6cbc0aa1e9425688158969?alias&admin"

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="CAStor administrator",
nonce="0c6da76911cbf5cd495afbb0c66e6d9a",
 opaque="3e894c0cf7c1ad980e1fd46320307f1a", stale=false, qop="auth",
algorithm=MD5
WWW-Authenticate: Basic realm="CAStor administrator"
Content-Length: 53
Content-Type: text/html
Date: Thu, 18 Apr 2016 15:13:00 GMT
Server: CAStor Cluster/9.1.0
Allow: HEAD, HOLD, GET, SEND, PUT, RELEASE, POST, COPY, GEN, APPEND,
DELETE

HTTP/1.1 200 OK
Castor-System-Alias: 97f7149dec6cbc0aa1e9425688158969
Castor-System-Cluster: baker
Castor-System-Created: Thu, 18 Apr 2016 15:10:40 GMT
Castor-System-Version: 1366297840.592
Content-type: text/xml
Last-Modified: Thu, 18 Apr 2013 15:10:40 GMT
transfer-encoding: chunked
Etag: "166e93908fc32ffb5f55beb7ed531ba1"
Volume: 8f61a5127994365e3dd89bbf83aa0964
Volume-Hint: b79cf7801f71f545c62957ae5659299b
Date: Thu, 18 Apr 2016 15:13:01 GMT
Server: CAStor Cluster/9.1.0

Remote cluster returned: 201
```

SCSP UPDATE

This section provides general information about SCSP UPDATE that applies to both named and unnamed objects. The UPDATE request is formatted as a simple HTTP request using the PUT method.

SCSP Method	HTTP Method	RFC 7231 Section
UPDATE	PUT	4.3.4

Special Query Arguments

replicate	Protects rapid updates	Important: Objects can be updated at a maximum frequency of once per second. Updating more frequently can cause unpredictable results with the stored object version and can trigger a 409 (Conflict) error. If your application updates objects faster than once per second, include the `replicate=immediate` query argument to ensure that more than one node can return the latest version in a subsequent read.
newname	Renames object	To rename a named object within the same bucket, use the newname query argument, which provides a new name with the update request (PUT, COPY, APPEND). After you rename an object, requests for the original name return a 404 Not Found and the prior search metadata is removed. (Note that the `newname` argument also lets you rename domains and buckets.)
preserve	Updates custom headers	PUT only saves new headers, but the `preserve` argument lets you keep the existing headers as well as save any new ones. (v9.5)

UPDATE for named objects

UPDATE is a request to the storage cluster to modify a specific named object or alias object with new content. The UPDATE request is formatted as a simple HTTP request using the PUT method:

```
PUT /bucket/file.txt HTTP/1.1
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
  Content-Length: 43402
  Expect: 100-continue
  Content-Type: image/jpeg
  Content-Language: en/us, x-pig-latin
  Content-Version: 42
  Last-Modified: Wed, 1 Sept 2010 15:59:02 GMT
  Created-Date: Wed, 1 Sept 2010 15:59:02 GMT
  CRLF
  [ content ]
```

UPDATE for unnamed objects

The UPDATE request is formatted as a simple HTTP request using the PUT method. The normal response to a PUT request, similar to a POST, is a 201 Created response.

```
PUT /06eec5e2c3f1aadcb41ef7fd52adc049 HTTP/1.1
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
  Content-Length: 43402
  Expect: 100-continue
  Content-Type: image/jpeg
  Content-Language: en/us, x-pig-latin
  Content-Version: 42
  Last-Modified: Wed, 1 Sept 2010 15:59:02 GMT
  Created-Date: Wed, 1 Sept 2010 15:59:02 GMT
  CRLF
  [ content ]
```

PUT returns a 404 Not Found error if the object name or UUID you specify in the command does not exist.

> Note
> If a non-erasure-coded object is updated and the update causes the object to meet the criteria described in Erasure Coding, the object can be erasure-coded, which has a smaller storage footprint.

UPDATE for alias objects

To update alias object, adding alias=yes is optional because this method applies only to mutable objects.

If UPDATE succeeds on an alias object, the content sent in the body of the request will be written as a new object. The first line of the HTTP PUT will be updated to point to the new object, and the original UUID of the object is returned to the client.

Normal Responses to UPDATE

> For a list of response headers, see SCSP Headers.

Error Responses to UPDATE

If you execute the UPDATE method on an object in a domain but the domain does not exist or is not in the content cache on the node that receives the request, Swarm responds with 409 Conflict.

> Note
> Rapid updates of an object can trigger a 409 Conflict error, that a "Later version already exists."

SCSP WRITE

- WRITE for named objects
  - Using PUT Create for named objects
  - Preventing overwriting: If-None-Match
- WRITE for unnamed objects
- WRITE for alias objects
- WRITE for erasure-coded objects
- WRITE for large files (Expect: 100-continue)

This section provides general information about SCSP WRITE that applies to both named and unnamed objects.

WRITE is a request to the storage cluster to create a new object. The WRITE request is formatted as a simple HTTP request using the POST method.

SCSP Method	HTTP Method	RFC 7231 Section
WRITE	POST	4.3.3

> Write for contexts
> The Swarm setting scsp.requireExplicitContextCreate protects content-bearing objects from being created erroneously as contexts (buckets or domains). With this setting enabled, Swarm will not create a context object unless it includes the required header: Content-type: application/castorcontext. (v9.1)

> S3 compatibility
> The Swarm setting scsp.autoContentMD5Computation improves S3 compatibility by automating Content-MD5 hashing, which means that you do not need to include the gencontentmd5 query argument or the deprecated Expect: Content-MD5 header on writes (although you may want to supply your own Content-MD5 header for content integrity checking). This setting is ignored wherever it is invalid, such as on a multipart initiate/complete or an EC APPEND. (v9.1)

WRITE for named objects

If you WRITE a named object that already exists, the existing object is overwritten with a new version.

WRITE that overwrites object

```
POST /bucket/photo.jpg HTTP/1.1
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
  Content-Length: 43402
  Expect: 100-continue
  Content-Type: image/jpeg
  Content-Language: en/us, x-pig-latin
  Content-Version: 42
  CRLF
  [ content ]
```

To prevent overwriting an existing object, include the If-None-Match: * request header.

- If the named object does not exist, Swarm WRITEs the named object.
- If the named object exists, Swarm responds with a 412 Precondition Fail error.

WRITE that prevents overwriting

```
POST /bucket/photo.jpg HTTP/1.1
  If-None-Match: *
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
  Content-Length: 43402
  Expect: 100-continue
  Content-Type: image/jpeg
  Content-Language: en/us, x-pig-latin
  Content-Version: 42
  CRLF
  [ content ]
```

Using PUT Create for named objects

If you want to configure Swarm to allow you to use the HTTP PUT operation to create new named objects, add the scsp .allowPutCreate=True to your configuration parameters. You can also enable it using the putcreate query argument.

Exception

Although domains and buckets are named, Swarm processes all PUT requests on these objects as updates, regardless of the setting.

- If the putcreate=yes query argument is used on a domain or bucket, Swarm fails the request with a 400 Bad Request error.
- If the scsp.allowPutCreate parameter is enabled, Swarm silently ignores it and processes the request as an ordinary PUT.

Preventing overwriting: If-None-Match

In contrast to an unnamed object, if you WRITE a named object that already exists, the existing object is overwritten with a new version. To prevent overwriting an existing object, include the If-None-Match: * request header.

- If the named object does not exist, Swarm WRITEs the named object.
- If the named object exists, Swarm responds with a 412 Precondition Fail error.

Note

Swarm returns a 412 error on SCSP WRITE of named objects if the domain or bucket does not exist or cannot be loaded.

Note

If-None-Match:* can erroneously report that an object exists during the time window after it is flagged for deletion by policy but before it is removed from disk. This window is determined by the HP cycle time.

WRITE for unnamed objects

Swarm makes no assumptions about User-Agent (except that it is an HTTP/1.1 client). The Host header must conform to the requirements of Section 14.23 of the HTTP/1.1 spec.

WRITE unnamed to host domain

```
POST / HTTP/1.1
  Host: cluster.example.com
  User-Agent:Swarm Client/0.1
  Content-Length: 43402
  Expect: 100-continue
  Content-Type: image/jpeg
  Content-Language: en/us, x-pig-latin
  Content-Version: 42
  CRLF
  [ content ]
```

If you WRITE an unnamed object, a new object is created and a new UUID is returned. If you WRITE an alias object, a new object is created and a new alias UUID is returned.

> See WRITE for Unnamed Objects.

WRITE for alias objects

To create alias objects, add alias=yes:

WRITE for alias object

```
POST /?alias HTTP/1.1
  Host: cluster.example.com
  User-Agent: Swarm Client/0.1
```

WRITE for erasure-coded objects

A new object written to the storage cluster is erasure-coded if it meets the EC criteria language.

> See Working with Large Objects.

Note
If you WRITE an object of more than 4 TB in size, Swarm returns a 503 Service Unavailable error.

WRITE for large files (Expect: 100-continue)

The Expect: 100-continue header tells the server that the client will wait after sending just the header lines and before sending the content in the message body. The Swarm server can respond with a redirect or an error response.

- If the server is ready to store the contents, it returns a 100 Continue response, telling the client to transmit the entity body. The client should wait for a 100 Continue response from the server before proceeding to send the data.
- If the server is not ready, it sends a 413 Request entity too large error response and closes the connection.

Swarm allows the client to omit the Expect: 100-continue header, sending all the content at once. In response, the server reads and discards all data if it must respond with a redirect or error. For WRITE messages that include more than 65536 bytes, Swarm logs a warning:

Error if Expect header is missing

```
Please use Expect: 100-continue for large amounts of data.
```

If any node in the cluster does not have enough space to write the object, the cluster returns a 507 Insufficient Storage error.

- WRITE with Replicate ROW

WRITE with Replicate ROW

The Replicate on Write (ROW) option forces Swarm to write a new object to one or more additional nodes before returning a success status. Using this content protection option, you can ensure that two or more object replicas (instances) exist in the cluster before the client write request is completed.

Success conditions for ROW

These are the success conditions for a ROW request:

- A POST on an immutable object creates at least two replicas.
- Any write operation for an alias object, or a POST for a named object.

> Note
>
> The reason for treating named objects like existing alias objects is that they might already exist. Allowing these writes to succeed with one replica ensures that no old versions can be inadvertently deleted by the HP should the request fail with only one replica.

Implementing ROW

You implement ROW in these ways:

- Globally, set the configuration file to enabling (recommended) or disabling cluster-wide ROW.
  - See Configuring Replicate On Write.
- Programmatically, use a replicate query argument whenever you need to override the cluster-wide ROW configuration.
- Creating or updating a bucket. The replicate=immediate option quickly invalidates cached bucket versions in the cluster so that the latest version will be implemented in the cluster. It also prevents subsequent permission errors because out-of-date permissions are used from the prior version.

```
curl -i
 --post301
 --data-binary ''
 --location-trusted
'http://172.16.0.35/bucket?domain=test.example.com&replicate=immediate'
 -D create-bucket.log
```

replicate argument

To control how Swarm implements ROW on a given request, add the replicate query argument.

If you have cluster-wide ROW enabled (recommended), use this argument to limit or disable ROW for the request:

- replicate=x (where x is an integer) creates x replicas on write. For example, replicate=1 allows the write to succeed with only one instance of the object.

If you have cluster-wide ROW disabled , use these arguments to enable ROW for the request:

- replicate=immediate is replicate=2, which ensures that two replicas are written.
- replicate=full is replicate={# of reps specified by lifepoint, or else `policy.replicas default`}

In every case, the number of replicas Swarm makes synchronously on the request cannot exceed the number of replicas specified in the lifepoint (or, if none, `policy.replicas default`). For example, for an object with no lifepoint specified, in a cluster with default=2, making a request with replicate=3 will still only cause 2 replicas to be synchronously created on the request.

Replica-Count header

Swarm indicates the number of replica created with the request in the Replica-Count header. To ensure you received the correct number of replicas, check the header value in the response.

If Swarm cannot locate at least two nodes in the cluster that will replicate the object, it will return a 412 Preconditioned Failed response. However, if Swarm can locate a PAN and one ROW peer node, it proceeds with the request.

Although sometimes a ROW request can return successfully with only one replica created, it will never attempt to do the operation if it cannot find at least two nodes up front.

Responses for replicating objects

If Swarm is replicating an object and the cluster cannot locate at least two nodes to store the replicas initially, it returns a 412 Preconditioned Failed response.

If Swarm locates one node to store the replica, it returns a 201 Created response. Applications that need to verify the requested number of created replicas should check the Replica-Count header value to verify how many replicas were created in the cluster.

If the requested number of replicas does not match the Replica-Count header value, repeat the request. Otherwise, the Health Processor will create the additional replicas at a later time.

To POST any unnamed object, Swarm locates two peer nodes—including the SAN—to perform the write. When two nodes are found and the writes are initiated, an immutable POST is considered a success if at least two replicas complete successfully. If Swarm cannot locate two peer nodes, the write fails and Swarm returns a 412 Preconditioned Failed response. All other writes are considered a success if at least one replica completes successfully.

WRITE for Unnamed Objects

You can create and run SCSP methods on unnamed objects in any domain. Housing (tenanting) unnamed objects in domains supports metered environments that need to allocate storage to users based on the domain. Unlike named objects, however, the domain is not used to later locate the unnamed object in the cluster.

For you to be able to create an unnamed object in a domain, your cluster administrator must have created the domain and enabled the cluster configuration setting cluster.enforceTenancy.

You write an unnamed object to a specific domain by including the domain in a query argument or in the HOST header:

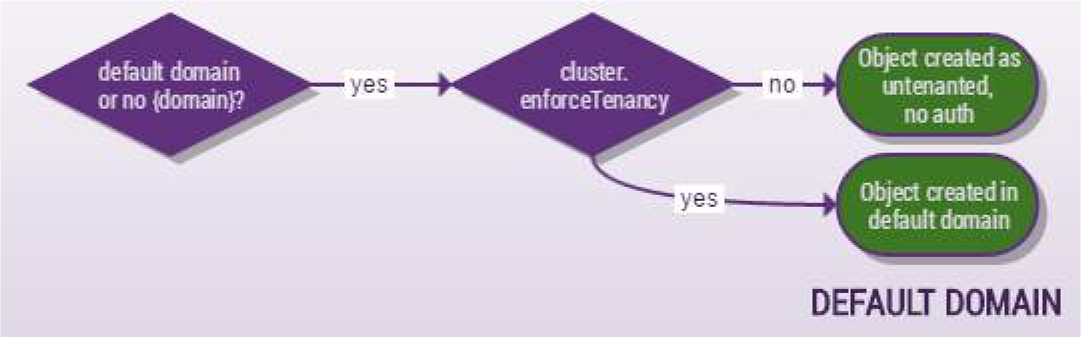
Which domain?	Alias Object	Immutable Object	

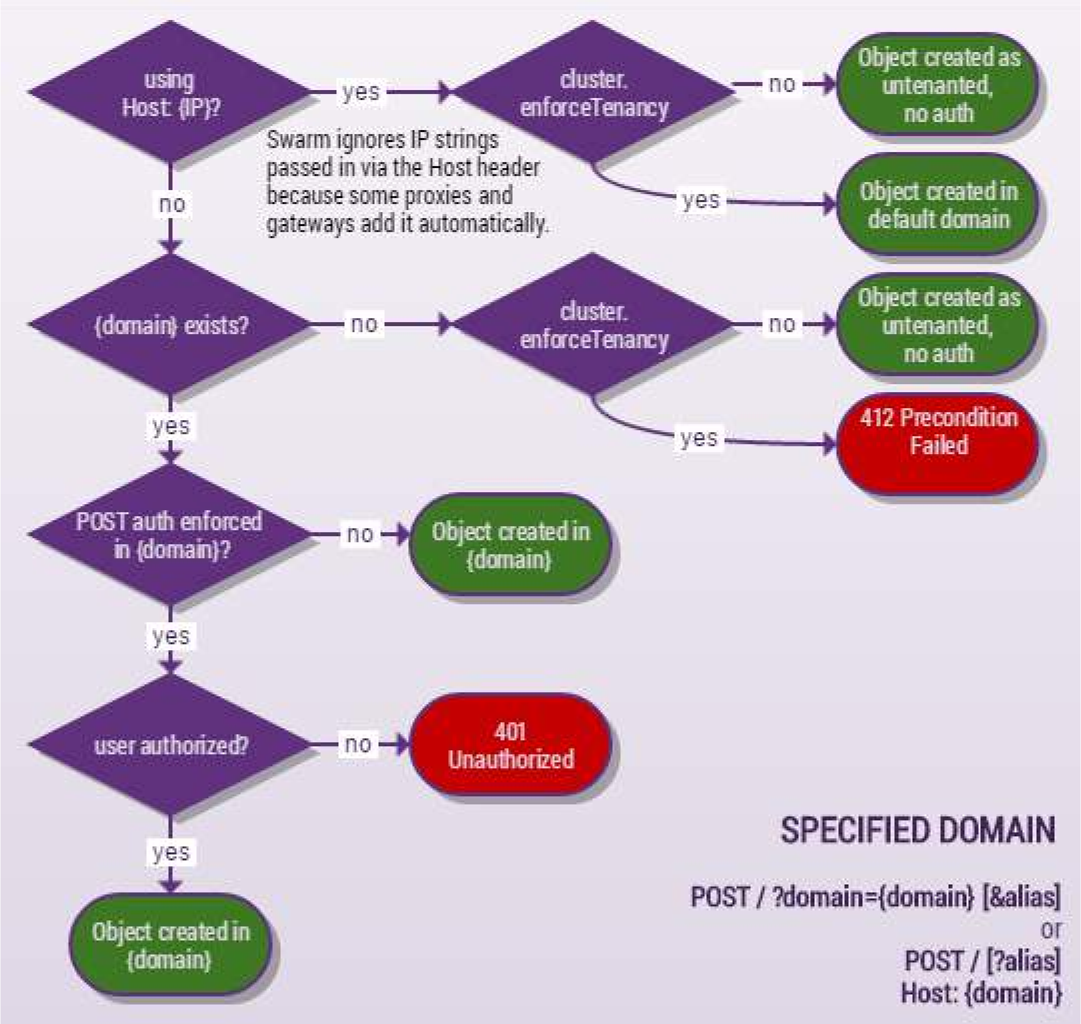
Unspecified (default domain)	POST /?alias	POST /	Every unnamed object that has no domain explicitly defined belongs to the default cluster domain. Ensure that the cluster administrator has set up a default domain, which is a domain name that exactly matches the name of the cluster.
By query argument	POST /?domain=domain-name&alias	POST /?domain=domain-name	
By host name	POST /?alias Host: domain-name	POST / Host: domain-name	Performance Warning If your application passes an invalid HOST header (a domain that does not exist in or match the cluster name), Swarm will make several tries to look up the invalid domain before timing out, on every request.

The effect of enforceTenancy

To create an unnamed object in a domain, Swarm checks the value of the cluster.enforceTenancy configuration setting and then performs a specific set of procedures, depending on whether the cluster administrator enabled cluster.enforceTenancy, which is disabled by default.

The following figure summarizes how the cluster.enforceTenancy setting affects the writing of unnamed objects:


default domain or no {domain}?
yes
cluster. enforceTenancy
no
Object created as untenanted, no auth
yes
Object created in default domain
DEFAULT DOMAIN


using Host: {IP}?
yes
cluster. enforceTenancy
no
Object created as untenanted, no auth
yes
Object created in default domain
Swarm ignores IP strings passed in via the Host header because some proxies and gateways add it automatically.
no
{domain} exists?
no
cluster. enforceTenancy
no
Object created as untenanted, no auth
yes
412 Precondition Failed
yes
POST auth enforced in {domain}?
no
Object created in {domain}
yes
user authorized?
no
401 Unauthorized
yes
Object created in {domain}
SPECIFIED DOMAIN
POST / ?domain={domain} [&alias]
or
POST / [?alias]
Host: {domain}

> Note
> Regardless of the value of the enforceTenancy setting, no domain specification is needed or recognized for GET, HEAD, or DELETE requests on unnamed (alias or immutable) objects.

Normal Responses to WRITE

> For a list of response headers, see SCSP Headers.

The responses listed here can be returned by the storage cluster when the new object can be created as requested. The content could be created and written to the cluster by the node that receives the request or redirected to another node in the cluster.

Note
The `Replica-Count` header indicates the number of synchronous replicas that were created, if the number is greater than 1.

201 Created	The response below indicates the storage cluster has stored at least one copy of the supplied object other nodes can store additional replicas at a later time. The value of the Location header provides t newly-created object can be accessed in the cluster. The last portion of the URI is the symbolic name object.
	For unnamed objects, this value is repeated as the value of the Content-UUID header field. Note that Content-Type and Content-Length headers refer to the message payload of the response (if any).
	`HTTP/1.1 201`
	`  Created Date: Wed, 1 Sept 2015 15:59:02 GMT`
	`  Server: CAStor Cluster/5.0.0`
	`  Location: http://node-ip/bucket/name-or-uuid`
	`  Content-UUID: 41a140b5271dc8d22ff8d027176a0821`
	`  Content-Type: text/html`
	`  Content-Length: 68`
	`  CRLF A new object has been created as requested.`
	`  Its URL is http://node-ip/bucket/name-or-uuid`

202 Accepted	The response occurs for certain types of writes, such as a multi-part completion. It means that the re accepted, and it is pending. The final status will be returned both in the body, and as a trailing heade `-system-result`. After a 202, the request may still fail.

301 Moved Permanently	The response indicates the request has been redirected to another node, and should be retried there. requests of this storage cluster should be made through the new access node until another 301 resp received. There is no message-body, so the content length is always 0. The value of the `Location` he which node in the cluster receives the redirect.
	The client is expected to send another POST request using the exact URI contained in the Location h the auth= query argument.
	`HTTP/1.1 301 Moved Permanently`
	`  Date: Wed, 1 Sept 2015 15:59:02 GMT`
	`  Server: CAStor Cluster/5.0.0`
	`  Location:  http://node-ip/bucket/name-or-uuid?auth=value`
	`  Content-Length: 0`

307 Temporary Redirect	The response below is similar to the 301 response, except that the client should continue to use the (the one generating this response) for future requests until further notice.
	`HTTP/1.1 307 Temporary Redirect`
	`  Date: Wed, 1 Sept 2015 15:59:02 GMT`
	`  Server: CAStor Cluster/5.0.0`
	`  Location:`
	`http://node-ip/bucket/name-or-uuid/?auth=B1E1509329C7A5DD90DCF6`
	`  Content-Length: 0`

Error Responses to WRITE

Errors are accompanied by three error headers (v9.1):

- `CastorSystemErrorCode` - The request error code (if applicable). This code is usually a 4xx or 5xx HTTP response code, but it may not match the response code on the request.
- `CastorSystemErrorText` - The request error description (if applicable). Provides a human-readable description of the error.
- `CastorSystemErrorToken` - A unique error token for the specific error path (if applicable). Provides an easily parsed token that uniquely identifies the error.

The storage cluster could return the following responses when the content length header does not match the actual content length, the specified content cannot be written to the cluster, or if there is a problem with the WRITE request itself.

400 Bad Request

The response below indicates a problem with the WRITE request, such as missing mandatory headers, invalid message body, or any other violation of HTTP/1.1 by the POST request. The reason for the error is included in the message body of the response.

```
HTTP/1.1 400 Bad Request
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Content-Type: text/html
  Content-Length: 24
  CRLF Host header is required.
```

411 Length Required

The response below indicates the WRITE request did not supply the actual content-length, which is required.

```
HTTP/1.1 411 Length Required
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Content-Type: text/html
  Content-Length: 93
  CRLF WRITE requests must include a Content-Length header
specifying
    the exact byte-length of the content to be stored.
```

412 Precondition Failed

The response indicates that the named object already exists or that the bucket or domain cannot be found.

507 Insufficient Storage

The response below indicates the WRITE request did not succeed because the storage cluster did not have sufficient resources to fulfill it. The specific resource constraint is described in the entity (message-body) of the response.

```
HTTP/1.1 507 Insufficient Storage
  Date: Wed, 1 Sept 2012 15:59:02 GMT
  Server: CAStor Cluster/5.0.0
  Content-Type: text/html Content-Length: 38
  CRLF Not enough disk space to store requested content.
```

Search Queries

Swarm Storage integrates with Elasticsearch to let client applications list and search the metadata on the objects being stored in the Swarm cluster. You access these operations by applying query arguments that are specific to search.

> For details on implementing and managing Elasticsearch, see Elasticsearch for Swarm.

Scope of Searching in Swarm

Swarm looks up objects by the underlying `contextid` so that you can query by context (domain/bucket) name as usual but always get correct query results even if a domain or bucket has been renamed. After this lookup, Swarm generates the final content query, which supports a wide range of functionality:

- Filter by value. You can filter any metadata field using equality checks, greater/less than comparisons, and wildcard matches.
- Filter by context. You can also narrow the search by filtering the context using equality checks, greater/less than comparison, and wildcard matches. To restrict search to a bucket, just include the bucket in the URL path.
- Filter by object type. You can add the `stype` argument to filter by one of these Swarm types: domain, bucket, na med, alias, immutable, unnamed (both alias and immutable), or all.
- Operate on metadata fields. You can perform AND and OR operations on the values of metadata fields to find the matching objects.
- Sort by value. Your sort specification can combine multiple metadata fields, including the context.
- Paginate large result sets with sort markers. You can apply markers when the context is sorted (ascending or descending) and in conjunction with markers for other metadata fields, when the sort specification includes multiple fields.
- Calculate disk usage. The `du` aggregation filters the results to calculate disk usage, inclusive or exclusive of object replicas in the cluster.
- Locate versions. If you are using versioning, you can use the `versions` argument to surface all of the historical versions of a single object or of all objects in the context.

- Search Query Arguments
- Metadata Field Matching
- Listing Operations
- Search Operations
- Search Examples

Search Query Arguments

- marker Argument
- stype Argument

Argument	Values	Example	Usage

format	json xml	&format=json &format=xml	Required. Requests a search query and specifie desired output format. This query arg is required to trigger a domain/bucke query. This format applies to the body portic the HTTP/1.1 response only and does affect the format of the response hea All search operations are available wi either format.
context	string	&context=example.com/abc_	Specifies a search filtered to the dom bucket given. Use for inequality and w matching. Best practice: Do not use for equality matching. To search a bucket, specify domain={domain-name} and include t bucket in the URL. Caution: Wildcard matching across m buckets can impact performance.
contextid	UUID	&contextid=80a4957…	Specifies a search filtered to the cont identified by the UUID given.
decorates	UUID	&decorates=a95780a4…	Use in a listing query to find any anno objects that might exist for a given ET earlier query result “hash”). See Meta Annotation and Listing Operations.
domain	domain-name	&domain=example.com &domain=	In a normal SCSP storage request, the required Host header is the assumed (or context) for the operation. Most HTTP/1.1 clients and browsers use th portion of the URL as the value for the header in their Swarm request. If the v the Host request header does not ma domain name, supply the domain arg to explicitly override it. A domain argu always has precedence over the Host in the HTTP/1.1 request. Set the value equal to nothing (empty to find unnamed objects that are not tenanted in any domain.
domains	none (ignored)	&domains	Used only for listing the domains witl cluster. No value is used with this arg

du	withoutreps |yes|true withreps	&du=withreps&size=0	Dynamically queries the disk usage fc domain or bucket context. With du, us ize=0 query argument (to shorten the by preventing the return of the reques and add any filters needed. The comp result value appears in the Castor-Sy Bytes-Used[-With-Reps] respons header. ● withoutreps sizes the content uploaded: it finds the sum of the l the unique objects being stored. ● withreps sizes the storage impa finds the underlying disk space in of the objects that are stored in a domain or bucket context, a value weighted by the expected number replicas in the cluster, which is an approximation of the data footpri the results. See Storage in Use for how to query tl storage in use for domains and bucke
field-name	any field name	&x-width-meta<=800 &x-height-meta<=600 &content-type=image/bmp&or &content-type=image/png	Filters results to those that match the operations on one or more specified f Multiple filters are joined by a logical to specify OR, add the &or argument. Supports standard comparison opera specific fields to filter the results retu ● a = b, equal to ● a <= b, less than or equal to ● a >= b, greater than or equal to This is an overloading of normal quer argument processing.
fields	Comma-separated list of field names, or all	&fields=name,content-length &fields=all	Replaces the default fields and specif a comma-separated list, which fields display in the output (see Metadata F Matching). Use fields=all on bucket and doma listing requests to return all of the ava fields on each record in the response.

marker	empty value or a comma-separated list of sorted column key values	&marker=&sort=tmBorn&size=50	Provides a mechanism to use multipl requests to receive a complete result Defaults to `&sort=name`. Used with the `size` argument to pagi large result sets. Use an empty key to a new search. Use the last sort key va the results on the next request to con pagination from that place on the nex request.
or	`yes/true` to activate	&or	When used with multiple field compa joins them with a logical OR (default i AND). Defaults to false.
prefix	string	&prefix=Q1 &stype=unnamed&prefix=93f &stype=unnamed&prefix=93f&domain=	Matches the string to the start (prefix object's name or UUID. The names of objects in the output list will all share same prefix string. Named objects are always contained a bucket, and the first "/" following th bucket name is the delimiter. Any add slashes following the first slash are p the object's name. That is, for `/phot pple.jpg`, where `photo` is the bucket could retrieve the object using `prefi` Unnamed objects require the addition argument `stype=unnamed` in order to on UUIDs. To find untenanted unnamed objects, include "`domain=`" (empty string).
size	integer	&size=5000	Constrains the number of results to b returned by the query. Defaults to 10 This argument is equivalent in purpos the "limit" or "max-keys" arguments i cloud protocols. Tip: Set size to `0` whenever you do no the actual listing, such as when using "du" query argument.

sort	Comma-separated list of field names, with optional modifier `:asc`|`:desc`	&sort=size:desc,name	Sets the sort order of one or more fiel with the first field sorting first. Sortin direction defaults to ascending (`:asc` specify descending (`:desc`), add the modifier to one or more of the fields. Important: Sorting exacts a computat penalty, so sort output only when nec
stype	Comma-separated list of Swarm types	&stype=unnamed	Specifies the type of objects being requested in a search. Defaults to `al` Valid values: `domain, bucket, nam alias, immutable, unnamed, all`
versioned	`true/false`	&versioned=true	Lets you filter updateable (named anc objects by versioning status, checkin whether versioning was enabled in th context at the time they were written. Checks the value of the Castor-System-IsVersioned heade (v10.0)
versions versions&prefix	none name of object	&versions &versions&prefix=logo.png	Requests a listing of all versions of al objects. Requests a listing of all versions of th named object (`prefix={objectName` See Object Versioning.

`marker` Argument

The **`marker`** argument provides a mechanism to retrieve a single result set using multiple HTTP requests. This is useful when a large result set is impractical to receive using one HTTP request. All marker operations work using a sorted result set. If a `sort` argument is not provided, the system implicitly uses **`sort=name`** . The `size` argument is used to control how many items are returned per HTTP request.

On the first request, **`marker=`** (without a value) is used to indicate that you are requesting the beginning of the result set. For subsequent requests, the last value(s) from the sort field(s) is used to indicate the last record that you received from the previous request.

> Tip
> When using **`tmborn`** as the marker, use a UTC date-time or Unix timestamp (float). The microseconds and time-of-day portions of a UTC date-time are both optional.

Marker Example

Consider the following set of object names in the bucket `dictionary`.

```
applaud
appoints
arches
basically
boardwalk
buffers
carpet
defender
```

Receive the first four items in the result set:

```
GET
/dictionary?format=json&domain=example.com&size=4&fields=name&marker=
[
   { "name": "applaud" },
   { "name": "appoints" },
   { "name": "arches" },
   { "name": "basically" }
]
```

The last sort field value from the first request is **`basically`** , so the next request to continue receiving the result set is as follows:

```
GET
/dictionary?format=json&domain=example.com&size=4&fields=name&marker=bas
ically
[
   { "name": "boardwalk" },
   { "name": "buffers" },
   { "name": "carpet" },
   { "name": "defender" }
]
```

Since the result set is now exhausted, a subsequent request yields:

```
GET
/dictionary?format=json&domain=example.com&size=4&fields=name&marker=def
ender [ ]
```

Sort and Marker Relationship

The marker value is the last value or values from the sort fields. The general way to say this is that if you use: `sort={field1},{field2},...,{fieldN}` then your marker will contain as many last field values in it as well:

```
marker={lastValueField1},{lastValueField2},...,{lastValueFieldN}
```

Both ascending and descending sort order is supported for each of the sort order fields.

Determining Completion

The set of objects in the cluster is continually changing; therefore, in a sense, list/search requests are never complete. However, there is a criterion for considering marker requests complete. The `Castor-System-Object-Count` response header returns the number of objects remaining to enumerate including the records retrieved by the current request. When the `CastorObject-Count` value is equal to the number of records retrieved by the current request, a client may consider the iterating request complete.

Markers across the Domain

A strategy for using markers to retrieve a result set of a search across the entire domain is to sort based upon a unique tuple of metadata fields that all objects will have. An example:

```
sort=context,name
```

> See Baseline Metadata Fields.

The `name` field is a UUID or an application assigned string depending upon the object type.

`stype` Argument

The stype argument lets you filter the search results to certain types of objects. These are the values that you can use:

stype	Description	Notes
all	(default) All object types	

bucket	Bucket objects	Must reside in a domain. When listing a domain, `stype=bucket` is assumed implicitly, as this is a common operation.
named	Content objects with an application-supplied name	Must reside in a bucket. You can search across buckets by specifying a domain and not including a bucket in the URL.
unnamed	Content objects with a Swarm-assigned UUID	Cannot reside in a bucket. Tenanted in a domain unless cluster.enforceTenancy is disabled.
immutable	Immutable unnamed objects	A subset of `stype=unnamed`
alias	Mutable unnamed objects	A subset of `stype=unnamed`

When using stype, take care to avoid making invalid requests, such as specifying stype=unnamed when searching within a bucket (which cannot contain unnamed objects). However, stype=named in a domain context is valid, even though named objects exist in a bucket, because that bucket is contained within the domain context as well.

To list the unnamed objects in the domain, use the unnamed, immutable, or alias value of stype to select the desired set. To find untenanted unnamed objects, include `domain=` set to nothing (empty string). In search and listing results, untenanted unnamed objects return an empty string for the domain field.

> Important
> Unnamed objects are only guaranteed to be tenanted in a domain when the cluster.enforceTenancy configuration option is enabled. To find untenanted objects, include `domain=` (empty string).

Metadata Field Matching

In addition to specifying query arguments, metadata field matching criteria are specified in the URI. This allows for fine-tuning the result set to return only objects in the storage cluster that match one or more matching criteria.

Matching criteria are logically AND expressions by default and can be switched to OR expressions using the or=yes query argument. The context for the search, everything after the domain name, and any value for the prefix argument are always considered to be logical AND constraints for the match. In other words, if a bucket name or a prefix pattern are specified in the URI, those must match even when or=yes is being used.

When matching, values can be specified using the "*" character to indicate glob style matching for multiple characters or the "?" character to indicate any single character for the field values. The following examples show the different matching concepts.

Matching examples

> Note
>
> `<cluster>` in a URL stands for `<host>[:<port>]`, where `host` is a fully qualified domain name or IP address, plus a `port` number if other than 80. If the Host header does not match the domain name, override it with the domain= argument.

Return only JPEG images

```
GET http://{cluster}/
  ?format=json
  &content-type=image/jpeg
```

Return JPEG or PNG images

```
GET http://{cluster}/
  ?format=json
  &or=yes
  &content-type=image/jpeg
  &content-type=image/png
```

Alternately, you could have used glob-style pattern matching:

```
GET http://{cluster}/
  ?format=json
  &content-type=image/*
```

Return JPEG or PNG images in a bucket
For example, {png OR jpeg} AND "pics" bucket:

```
GET http://{cluster}/pics
  ?format=json
  &or=yes
  &content-type=image/jpeg
  &content-type=image/png
```

Match a single positional wildcard
This example finds values including "grey" or "gray":

```
GET http://{cluster}/
  ?format=json
  &x-color-meta=gr?y
```

Searchable metadata fields

Each Search Feed indexes metadata for searching, but which metadata depends on how you define the feed in the Storage Management UI or Swarm Admin Console. With the Search full metadata checkbox selected, Swarm indexes all available metadata for the objects in the cluster; with it unselected, Swarm indexes only basic metadata fields to support listing operations.

if you implement Search full metadata on your search feed, allow for additional storage and RAM on the search servers to support it.

> Tip
> Even without Search full metadata enabled, you can still perform searching on the basic metadata fields.

Basic metadata fields

The following table provides a list of the standard, baseline field names as they are mapped between the name used in the query argument values and the name given in the XML and JSON output formats. Notice that the output name may be different from the name used in the query argument and that the output name can change depending upon the output format.

Query Arg	XML Name	JSON Name	Description
`tmBorn`	`LastModified`	`last_modified`	Time of create or last update. The query arg may use either UTC date-time or Unix timestamp (float) in search requests. The microseconds and time-of-day portions of a UTC date-time are both optional.
`content-length`	`Size`	`bytes`	Size in bytes

`name`	`Key`	`name`	UUID or name using URL encoding
`content-type`	`content-type`	`content_type`	Content type
`etag`	`ETag`	`hash`	Entity tag
`sizewithreps`	`sizewithreps`	`sizewithreps`	Number of bytes using the maximum reps value

Full metadata fields

The following list shows the metadata field names that are indexed when you license full metadata search. Custom Metadata Headers are included in these patterns.

- castor-* (except castor-system-*)
- content-base
- content-disposition
- content-encoding
- content-language
- content-location
- content-md5
- lifepoint
- x-*-meta[-*]

However, these fields do not appear in listings unless you explicitly include them in the fields query argument:

```
GET http://{cluster}/mybucket
 ?format=xml
 &fields=name,content-length,x-color-meta
```

> See Search Operations.

Case: While the metadata field names are case-insensitive for the purposes of matching, they are stored in the cluster as given during the WRITE operation. The metadata field values are case-sensitive.

Note
Custom metadata field names that contain hyphens (-) will have these characters converted to underscores (_) in the result output, but this conversion is only for the output text. Continue to use hyphens in the URI for specifying field names.

Listing Operations

- Listing domain contents
- Listing a bucket context
- Listing untenanted unnamed objects
- Storage in use

- Domain storage in use
- Bucket storage in use
- Annotations in existence

Listing operations are a specialized class of searching that usually have a context constraint of a domain or a bucket, except in the case of listing untenanted unnamed objects or finding annotation objects. When performing listing operations, the user is typically interested in the hierarchy or membership within a context. Since listing operations are in fact searches, you can use other searching options and metadata constraints in combination with them.

> Encoding
>
> When non-ASCII characters are included, list query response bodies are UTF-8 encoded. When reading and writing non-ASCII characters, applications must decode the response body from UTF-8 prior to interpreting the list body.

Listing domain contents

Domains are a context that contains bucket objects and unnamed objects. Buckets are identified by name. Unnamed objects are identified by UUID and are either mutable (alias) or immutable.

Basic form of listing a domain

```
GET /?format=json&domain=myDomain
```

Filter buckets that start with a string

```
GET /?format=json&domain=myDomain&prefix=Southwest_
```

Listing unnamed objects within a domain

```
GET /?format=json&domain=myDomain&stype=unnamed
```

Listing everything within a domain

```
GET  /?format=json&domain=myDomain&stype=all
```

> Note

Swarm assumes stype=bucket when listing a domain, so it returns a list of bucket names unless you request a specific stype.

Listing a bucket context

Buckets are a specific context type that belongs to a domain and can contain only named objects. This operation is similar to listing a domain context with additional URI element of the bucket name added.

Listing bucket contents

```
GET /myBucket?format=json&domain=myDomain
```

Note
Swarm assumes stype=named when listing a bucket because no other stype is valid.

Listing untenanted unnamed objects

If the storage cluster includes unnamed objects that are not contained in a domain (untenanted), you may still list those objects by specifying an empty domain context.

Listing untenanted unnamed objects

```
GET /?format=json&domain=&stype=unnamed
```

Storage in use

You can dynamically query the storage in use for a domain or bucket context using the du query argument. This is valid for domains and buckets.

- du=withreps requests the total storage impact of objects.
- size=0 prevents the return of the request body (the calculated value is returned in the header).

Argument	Header Result	Description
du=withoutreps	`CastorBytesUsed: 2189243`	For du=withoutreps, the HTTP/1.1 response header returns a Castor-System-Bytes-Used field that indicates a summary of the storage used by the objects within that context. Swarm calculates the space from the sum of the body bytes of all relevant and distinct objects. This calculation does not consider the number of replicas for each distinct object. .

du=withreps	`CastorBytesUsedWithReps: 109416625`	For du=withreps, the HTTP/1.1 response header returns a Castor-System-Bytes-Used-With-Reps field that indicates a summary of the storage used by the objects within that context. Swarm calculates the space from the sum of the body bytes of all relevant objects using each object's maximum reps value from the object lifepoint headers and the assigned value from the cluster multiplied by the object's size. A 100 MB object with reps=2 will consume 200 MB of space. The same object with reps=3 will consume 300 MB of space. Stored with an erasure coding value of reps=4:2, the object will consume 150 MB of storage.

If the object has three lifepoints that include all of these previous example values, the maximum of reps=3 will be chosen for the calculation and the storage impact will be recorded as 300 MB. Every object has the metadata field size withreps that records its space impact.

> Important
>
> The Castor-System-Bytes-Used and Castor-System-Bytes-Used-With-Reps fields are computed on-demand. Depending upon the object count within the context, they can consume computational resources on the search servers. Applications should only request du operations when the space calculations are required.

Domain storage in use

When querying the storage in use for a domain, you have the option of selecting the types of objects to consider using the stype argument.

Argument summary:

- du=withreps requests the full storage impact of objects.
- size=0 prevents the return of the request body (the calculated value is returned in the header).
- stype=named|unnamed|all selects the types of objects included in the calculation.

Space used by all content in domain

```
GET /?format=json&domain=myDomain&du=withreps&size=0&stype=all

HTTP/1.1 200 OK
Gateway-Request-Id: 26F809F67D883E6D
Content-Type: application/json; charset=utf-8
Castor-System-Object-Count: 17
Castor-System-Bytes-Used-With-Reps: 121590
[snip]
```

Space used by named objects

```
GET /?format=json&domain=myDomain&du=withreps&size=0&stype=named

HTTP/1.1 200 OK
Gateway-Request-Id: C6A8D293950C1FD5
Content-Type: application/json; charset=utf-8
Castor-System-Object-Count: 12
Castor-System-Bytes-Used-With-Reps: 121422
[snip]
```

Space used by unnamed objects

```
GET /?format=json&domain=myDomain&du=withreps&size=0&stype=unnamed

HTTP/1.1 200 OK
Gateway-Request-Id: 3D79D93B73A07E35
Content-Type: application/json; charset=utf-8
Castor-System-Object-Count: 2
Castor-System-Bytes-Used-With-Reps: 168
[snip]
```

Bucket storage in use

When querying the storage in use for a bucket, there will only be named objects within the bucket context. As a result, the stype argument is not required.

Argument summary:

- du=withreps requests the total storage impact of objects.
- size=0 prevents the return of the request body (the calculated value is returned in the header).

Calculating space used by named objects in a bucket

```
GET /mybucket?format=json&domain=myDomain&du=withreps&size=0

HTTP/1.1 200 OK
Gateway-Request-Id: 100601F9E31D5ECC
Content-Type: application/json; charset=utf-8
Castor-System-Object-Count: 1000
Castor-System-Bytes-Used-With-Reps: 22016
[snip]
```

Annotations in existence

To retrieve any annotation objects that may exist in the cluster for a given object, submit a listing query that sets the argument "decorates" equal to the ETag of the target object in question:

Listing annotation objects for given ETag

```
GET /?format=json&domain=&decorates=8c2c582c216a1f088c3652bced5a5f91
```

> See Metadata Annotation.

Search Operations

Search operations are an extremely powerful feature for locating content; they work on the metadata of the objects within the storage cluster. Searches use metadata matching constraints provided in the client request and return a list of objects that match those constraints.

Searches can take place across all objects within the domain or searches can be constrained to the context of a particular bucket. When full metadata search is enabled, you can use any custom metadata field value as a search constraint.

> **Best practice**
> Never apply a context filter redundantly in cases where Swarm filters by default, such as when searching for named objects in a bucket, buckets in a domain, or unnamed objects in a domain.

Performance Impact

Use care when designing searches that span entire domains. In particular, be aware that `context` (domain and bucket) names are looked up from their underlying `contextid`, so these domain-wide searches incur an additional performance penalty:

- Retrieving the `context` field
- Sorting on the `context` field
- Filtering on the `context` field

> Tip
> In general, be sure to check the performance impact whenever sorting and filtering across an entire domain.

Search Examples

Unless otherwise noted, all matching operations are string-based comparisons:

Searching within a domain

```
GET /?format=json&domain=myDomain
 &content-type=application/pdf
```

Searching within a bucket

```
GET /myBucket?format=json&domain=myDomain
 &content-type=application/pdf
```

Searching by multiple field matching

```
GET /?format=json&domain=myDomain
 &x-color-meta=red
 &content-type=image/png
```

Using fields= to return specific field names

```
GET /myBucket?format=json&domain=myDomain
 &fields=name,content-length
```

Using name.lower (case-insensitive)

Elasticsearch queries can be run with different case-sensitivity by using the correct form of the name field:

- name field: ES searches are case-sensitive, so searching `FOO` matches only `FOO`
- name.lower field: ES searches are case-insensitive (as if all values were lowercase), so searching `FOO` matches `FOO, Foo, foo`

The Swarm search setting, `search.caseInsensitive`, is specific to SCSP queries, versus querying ES directly. (v9.0)

Swarm Setting	Effect
`search.caseInsensitive = 1`	All name-based searches use the name.lower field, so that SCSP names searches are always case-insensitive.
`search.caseInsensitive = 0`	All name-based searches use the name field, and will therefore be case-sensitive.

> Important
>
> Custom metadata values are always indexed to be only case-sensitive or case-insensitive, depending on the value of `search.caseInsensitive`. If an index is built with the wrong setting, you must change the setting and build a new index.

You can see the difference by querying case-insensitive metadata for both name and name.lower and then displaying and sorting those fields:

```
curl -XGET ivory1:9200/mists/_search?pretty=true -d '{
   "query":{
      "term":{
         "name":"_administrators"
      }
   },
   "fields":[
      "name",
      "name.lower",
      "tmBorn"
   ]
}'
```

```
curl -XGET ivory1:9200/mists/_search?pretty=true -d '{
   "query":{
      "term":{
         "name.lower":"_administrators"
      }
   },
   "fields":[
      "name",
      "name.lower",
      "tmBorn"
   ],
   "sort":[
      "name.lower":"asc"
   ]
}'
```

Using content-length

The content-length field for objects is recognized as a numeric field and supports equality, less-than-equal-to, and greater-than-equal-to matching operators.

> Tip
>
> You can use ">=" or "=>" and "<=" or "=<" interchangeably.

Using content-length

```
GET /?format=json&domain=myDomain&content-length=1024
GET /?format=json&domain=myDomain&content-length<=1024
GET /?format=json&domain=myDomain&content-length>=1024
```

> Important
>
> The **`Content-MD5`** metadata field cannot be used as a search constraint, either alone or with other fields. Use it only in the output fields for a search.

Search Examples

- Queries for Buckets
- Queries for Named Objects
- Queries for Unnamed Objects

- Queries for Named and Unnamed Objects

These examples of how to search Swarm are demonstrated through curl.

> Note
>
> `<cluster>` in a URL stands for `<host>[:<port>]`, where `host` is a fully qualified domain name or IP address, plus a `port` number if other than 80. If the Host header does not match the domain name, override it with the domain= argument.

Queries for Buckets

Simple query with fields, not involving context

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size"
```

Simple query with fields, retrieving context as one of the fields

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context"
```

Retrieving context as one of the fields and sorting on context

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&sort=context:asc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,contextid,context&sort=context:desc"
```

Sorting on multiple fields, context, and name, in different orderings

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&sort=context:asc,name:asc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&sort=context:desc,name:desc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&sort=name:asc,context:asc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&sort=name:desc,context:desc"
```

Listing query, retrieving context as one of the fields, sorting on multiple fields with context

```
curl -i -X GET
"http://{cluster}/bucket1?format=json&domain=example.com
 &fields=name,size,context&sort=size:desc,context:desc"
```

Queries for Named Objects

Retrieving context as one of the fields

```
curl -i -X GET
"http://{cluster}?format=json&domain=example.com
 &fields=name,size,context&stype=NAMED"
```

Retrieving context as one of the fields and sorting on context

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=context:asc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=context:desc"
```

Sorting on context and another field in different orderings

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=context:asc,size:asc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=context:desc,size:desc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:asc,context:asc"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc"
```

Sorting and an equality context filters

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &context=example.com/bucket1"
```

Sorting and inequality context filters

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &context>=example.com/bucket2"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &context<=example.com/bucket2"
```

Sorting and wildcard context filter

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &context=example.com/bucket*"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &context=example.com/bucket1*"
```

Sorting and context marker and multiple markers

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,siz,context&stype=NAMED&sort=context:asc
  &marker=example.com/bucket2"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:asc,context:asc
  &marker=15,example.com/bucket1"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=context:asc,size:asc
  &marker=example.com/bucket1,15"
```

With du argument, withreps and withoutreps

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &du=withoutreps"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &du=withreps"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,size,context&stype=NAMED&sort=size:desc,context:desc
  &context=example.com/bucket1&du=withreps"
```

Queries for Unnamed Objects

With context sort and filter on context

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,tmborn,context&stype=IMMUTABLE&sort=context:asc
  &context=example.com"
```

Queries for Named and Unnamed Objects

With context sort

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,tmborn,context&stype=IMMUTABLE,NAMED&sort=context:asc
  &du=withreps"
```

With context sort and context wildcard filter

```
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,tmborn,context&stype=IMMUTABLE,NAMED&sort=context:asc
  &context=example.com*&du=withreps"
curl -i -X GET
 "http://{cluster}?format=json&domain=example.com
  &fields=name,tmborn,context&stype=IMMUTABLE,NAMED&sort=context:asc
  &context=example.com/*&du=withoutreps"
```

Metadata Headers

Lifepoint Metadata Headers

You can use optional lifepoint headers to define object-specific Swarm replication and retention policies, which can be as simple or complex as your situation requires.

> See SCSP Headers.

Understanding Storage Policies

Each node in a storage cluster includes a Health Processor that continuously cycles through the list of content objects that it stores on disk to determine what is considered "healthy" for each object at this particular point in its lifecycle. For example, the Health Processor may determine that an object needs to have at least three replicas of itself stored within Swarm. This requirement referred to as a content constraint or simply a constraint enables the Health Processor to take the appropriate action when needed to ensure disk-level and lifecycle data protection.

You can specify a constraint when you first store the object in the storage cluster. For mutable or named objects, the constraint can be changed with a COPY or a PUT.

Constraints can also be grouped together and given an expiration date. This type of constraint group is called a lifepoint because it represents a point where the health requirements of an object will change. When you create a sequence of lifepoints, they are collectively called a storage policy or a content lifecycle.

Lifepoints to prevent deletion

Important

Allow headers have no effect on automatic deletes specified in Lifepoint headers. For best protection from deletes, use `deletable=no` lifepoints. Using lifepoints lets you block recursive deletes when a bucket or domain is deleted, causing Swarm to log a CRITICAL error that non-deletable content is present.

An important use of lifepoints is to protect objects from deletion. However, deleting a bucket that contains such protected objects will generate errors and orphan those named objects.

Best practice

If you want to maintain a bucket for undeletable objects, make the bucket object itself undeletable.

> See "DELETE for domains and buckets" in SCSP DELETE.

Lifecycle evaluation example

Assume that an object was written to Swarm on June 12, 2015. In the first six months of its life, the object must have at least three replicas and cannot be deleted by any user. In the second six months of its life, the object needs just two replicas, and client applications can delete the object. After a year, the object is deleted.

Complete lifecycle policy

```
Lifepoint: [Wed, 12 Dec 2015 15:59:02 GMT] reps=3, deletable=no
Lifepoint: [Sun, 08 Jun 2016 15:59:02 GMT] reps=2, deletable=yes
Lifepoint: [] delete
```

Note

If there is one replica of an object in a cluster, then there is one instance of the object. Replica and instance are synonymous in this context.

Each time the Health Processor (HC) examines the object, it checks the current date to see how to apply the lifepoint policies:

Timeframe	Lifepoint Effects	Notes

Before the first lifepoint date	Swarm refuses SCSP DELETE requests. HP maintains at least three replicas of the object in the cluster.	
Between the first and second lifepoint dates	Swarm accepts SCSP DELETE requests. HP allows the number of replicas in the cluster to decrease.	Now the lifepoint specifies the deletable constraint enables a client to delete the content by sending an SCSP DELETE message with the object's name or UUID
After the second lifepoint date	Swarm accepts SCSP DELETE requests. HP deletes the object at the first checkup.	Whenever the last lifepoint has no end date, it is in effect indefinitel y once it comes in range.

Specifying Lifepoints and Lifecycles

You can use a simple syntax to specify a complete object lifecycle, and you can specify one or more lifepoints. You do this by attaching lifepoint entity headers to an SCSP WRITE message.

The entity header is shown below in Augmented Backus-Naur Form (BNF) syntax:

```
lifepoint = "lifepoint" ":" end-date 1#constraint end-date = "["
[HTTP-date] "]"
 constraint = replication-constraint | delete-constraint |
deletable-constraint replication-constraint =
  "reps" ["=" (1*DIGIT | 1*DIGIT:1*DIGIT)] delete-constraint = "delete"
["=" ("yes" | "no")]
   deletable-constraint = "deletable" ["=" ("yes" | "no")]
```

Guidelines for lifepoints

When you create a lifepoint, follow these guidelines:

Guideline	Explanation

Make every lifepoint stand alone	Lifepoints do not build upon one another: they stand alone as a complete specification of the constraints that apply to the object in a given date range. Be sure to i nclude the complete set of constraints for a given end date in the lifepoint header. Correct lifepoint `Lifepoint: [] reps=1,deletable=no`
Give time in GMT	For HTTP-date, adhere to the Full Date Section 3.3.1 of the HTTP/1.1 specification. This means that the indicated time must be specified in Greenwich Mean Time (GMT). When dealing with Swarm, GMT is exactly equal to UTC (Coordinated Universal Time).

Do not use deletable= without reps=	The delete constraint does not store a value and cannot include end-date :
	Incorrect delete constraint
	`Lifepoint: [] reps=1`
	`Lifepoint: [] deletable=no`

Do not delete contexts by lifepoint	To protect content objects from being orphaned, Swarm does not allow lifepoint-triggered deletes of contexts (domains and bucket objects). See SCSP DELETE for guidance on deleting domains and buckets.

Do not replicate chunked uploads	Chunked uploads are erasure-coded automatically, so a request will fail if it is chunked and the current lifepoint specifies replication.
	To convert a chunked upload, specify two lifepoints: have the first specify an EC encoding that expires in one day, and have the second specify the number of replicas that you want going forward:
	Converting chunked to replication
	`Transfer-Encoding: chunked`
	`Lifepoint: [Wed, 12 Dec 2016 15:59:02 GMT] reps=5:2`
	`Lifepoint: [] reps=3`

Do not expect Swarm to validate lifepoints	To maximize performance, Swarm does not validate lifepoints when they are added to the cluster. Swarm accepts an invalid lifepoint and later logs an error only if the HP cannot parse the lifepoint.

Constraints for Replication and Deletion

Constraint names and values are parsed by Swarm object classes called ConstraintSpecialists that maintain one or more related constraints. For example, the reps constraint is parsed and maintained by the ReplicationConstraintSpecialist. In general, constraint names are case-sensitive, and constraint names not recognized by any of the ConstraintSpecialists are ignored. As a result, the set of allowable constraints is extensible, and new constraint types may be added to the system in future releases.

Constraint names and arguments recognized by the ConstraintSpecialists in Swarm include:

- ReplicationConstraintSpecialist
- DeletionConstraintSpecialist

ReplicationConstraintSpecialist

The ReplicationConstraintSpecialist maintains the desired level of redundancy of content objects and ensures they are stored in the most efficient manner. It understands one constraint name: reps, which is set by protection type:

- Replicas – a single integer value
- EC – a tuple of k:p integers (such as `5:2`)

The ReplicationConstraintSpecialist does this by ensuring that the actual number of replicas or segments for an object is equal to reps at all times. If a replication constraint is missing from the lifepoint, a default value is supplied from the node or cluster configuration. Cluster administrators have control over some aspects of replication behaviors through Swarm configuration parameters:

- Replicas – Place limits on the number of replicas that can be specified by defining policy.replicas min and max.
- EC – Specify the ec.minParity to ensure that all objects have a minimum number of parity segments included for protection. If invalid or conflicting values of the reps constraint are found in a lifepoint, they are ignored, defaults are used, and warnings are written to the log. Lifepoints with erasure coding define what EC level to apply. For example: lifepoint = [] reps=5:2 expresses an erasure-coded level of 5 data segments and 2 parity segments.

Supported conversion methods

As of v6.5, a storage policy with multiple lifepoints that include the following conversion methods are supported:

- Replication to EC
- EC to replication
- One EC encoding to a different encoding

> Important
>
> The object size value must be greater than the policy.ecMinStreamSize setting, regardless of the specified lifepoint. Otherwise, the object will not be erasure-coded and will instead be protected with p+1 replicas.

DeletionConstraintSpecialist

The DeletionConstraintSpecialist completely removes a content object at a certain point in time and allows or disallows client applications to delete the content object using the SCSP DELETE request.

DeletionConstraintSpecialist understands two constraint names: deletable and delete.

- The deletable constraint is set to `yes|true` or `no|false`:
  - `yes|true` (default) indicates that the object is deletable by any client that knows its name or UUID. The DELETE method must also be included in the Allow header for a client delete to be allowed.
  - `no|false` prevents any agent from deleting the object during the effective period of the lifepoint. Any attempt to delete the object result in a 403 (Forbidden) response.
- The delete constraint does not accept a value. This constraint causes DeletionConstraintSpecialist to delete the content object from the cluster. The result is the same as if a client application had deleted the object.

To avoid ambiguity, when delete is present in a lifepoint specification, it must be the only constraint in that lifepoint because other conditions on a deleted object may not be applicable. Additionally, a delete lifepoint must be specified with an empty end date.

Incorrect delete constraint

```
Lifepoint: [Wed, 08 Jun 2012 15:59:02 GMT] reps=3, deletable=no, delete
```

Correct delete constraint

```
Lifepoint: [Fri, 12 Dec 2011 15:59:02 GMT] reps=3, deletable=no
Lifepoint: [] delete
```

Note
Do not use deletable=no and delete in the same lifepoint.

Allow Metadata Header

- Allow for alias objects
- Administrative Override
  - Evaluating success

The HTTP Allow entity header is used to specify which HTTP methods can be executed for an object.

Important
Allow headers have no effect on automatic deletes specified in Lifepoint headers. For best protection from deletes, use `deletable=no` lifepoints. Using lifepoints lets you block recursive deletes when a bucket or domain is deleted, causing Swarm to log a CRITICAL error that non-deletable content is present.

> See SCSP Headers

Allow for alias objects

The GET and HEAD methods are always supported (regardless of the Allow header), so there is no need to include an Allow header on an unnamed object. For alias objects, the Allow header can meet several use cases. For example, if you want an alias object to be mutable for a short time, you can use the following header with a PUT request when the user is ready for the object to become immutable:

```
Allow: GET, HEAD
```

This removes PUT, COPY, and APPEND from the supported methods, effectively making the object immutable.

When asked to perform a request on an alias object, the SCSP server will first examine its metadata for the presence of an Allow header. If found, it will return a 405 - Method not allowed response for any method not found in the list. The error response includes the Allow header stored with the alias object to provide guidance to the application about which methods are allowed for this object. If no Allow header is stored with the alias object, the default result is to allow all methods except POST for alias objects.

An alias object is dynamically deletable only if the DELETE is included in the Allow header and its current lifepoint allows deletes (deletable=yes). The Allow header has no effect on automatic deletes specified in lifepoint headers, which cause an object to be deleted at a certain point in time. Such lifepoint deletes do not require executing a DELETE method, and therefore do not contradict any Allow header that may not include support for deletes.

Administrative Override

To prevent content from being stranded with no ability to delete or update it due to an overly restrictive Allow header, Swarm supports administrative override of the Allow header.

> Tip
>
> The primary use for the override is to update the Allow header using a COPY request to enable additional SCSP methods that are needed.

To apply the administrative override, use the query argument **`admin[=yes|true]`** with the request. **`admin`** with no optional argument defaults to **`true`**. Any value other than yes or true is interpreted as false and the administrative override request is ignored.

The **`admin`** query argument indicates that Swarm should evaluate the request for administrative authorization. In addition to including the admin query argument, you must be in the CAStor administrator user list and include your credentials with the request in a standard HTTP Authorization header, as defined by the HTTP/1.1 spec and the corollary HTTP Authentication specification.

Example of an Authorization header:

```
Authorization: Digest username="JoAdmin",
 realm="Castor administrator",
 uri="/bucketname/objectname",
 response="credentials_digest"
```

- Bad credentials: If the administrative request does not include an Authorization header with suitable administrative credentials, Swarm will respond to the request with 401 Unauthorized, which includes a WWW-Authenticate challenge containing the administrative domain named Castor administrator and other required items.
- Good credentials: If the request includes both the query argument and authorized administrator credentials, it proceeds and the Allow header is ignored.

Administrative overrides cannot be used for methods that are never supported for an object, specifically update methods like PUT, COPY, or APPEND to immutable objects. If immutability might need to be overridden in the future, consider writing the object as an alias object with an Allow header that does not include any of the update methods: This prevents normal users from modifying the object but allows the administrator to update it using an authorized administrative request, if needed.

For audit purposes, all administrative requests are logged along with the user name of the requester.

Evaluating success

To determine whether a particular SCSP method succeeds, Swarm examines the following in order:

1. The `admin` query argument, which, if present, bypasses other authorization methods.
2. The methods allowed by the `Allow` header.

Custom Metadata Headers

- Requirements for Custom Names
- Requirements for Custom Values
- Sample Scenario for Custom Metadata

You can create custom metadata headers as a means to pass data required by your application. Including custom metadata on stored objects increases the usefulness of your content: it provides information that can be indexed by Elasticsearch and used to find, filter, and analyze the content later.

> Note
> Swarm stores these headers and their supplied values without parsing, validation, or modification.

You work with custom metadata through the WRITE, UPDATE, and COPY methods. The COPY method lets you update and add to the metadata on objects after the initial WRITE.

> Tip
> With COPY requests, you can add the preserve query argument to ensure that any custom metadata existing on the object is carried over to the copy. To overwrite an existing value, include the header name with the new value on the request. (v9.2)

> See SCSP Headers.

Requirements for Custom Names

- For best compatibility going forward, Swarm restricts you to these characters in your custom metadata header names (v9.1):
  - letters (both cases, although case-insensitive is consistent with HTTP/1.1 RFC)
  - numbers
  - dash (hyphen)
  - underscore

  > Important
  > Elasticsearch 2.3.3 does not allow dots in normal field names. When indexing objects for Elasticsearch 2.3.3, Swarm converts any legacy dots in custom metadata field names (`x_foo_meta_2016.12`) to underscores (`x_foo_meta_2016_12`). (v9.1)

- Follow one of these two naming formats when you define custom headers, or they will be silently ignored and not persisted to the storage cluster:
  - x-*-meta
  - x-*-meta-*

  ```
  x-ExampleCorp-meta-color: blue
  ```

Requirements for Custom Values

- To specify more than one value for the same header, list the values on the same line, separated by commas.

  ```
  x-color-meta: blue, green
  ```

  > Important
  > Do not reuse the same header with different values.

- For metadata values, use 7-bit US-ASCII characters, or else follow RFC 2047 guidelines for alternate character sets.

  ```
  x-xml-meta-data:
  <size>large</size><color>blue</color><specialorder/>
  ```

- Ensure that the total length of all persisted metadata, keys and values, does not exceed 32 KB. Metadata over 32 KB results in a 400 Bad Request error response from Swarm.

Sample Scenario for Custom Metadata

Assume a domain of "`example.com`" with a bucket called "`surveillance`", created for storing the company's surveillance videos.

To add a video, POST to the bucket, specifying the Content-Type of the video and including custom metadata to document the video's duration, camera location, and camera model:

```
curl -i --location-trusted -X POST --post301 \
 --data-binary @20170311-972-9928817883.mp4  \
 -H "Expect: 100-continue"      \
 -H "x-example-meta-Start-Time: 2017-03-11T12:00:01.678Z" \
 -H "x-example-meta-End-Time: 2017-03-11T13:00:00.421Z"  \
 -H "x-example-meta-Building: Annex 2" \
 -H "x-example-meta-Location: 972" \
 -H "x-example-meta-CameraModel: SWDSK-850004A-US" \
 -H "Content-Type: video/mp4" \
 -H "Content-Disposition: inline" \
 "http://example.com/surveillance/2017/03/22/20170311-972-9928817883.mp4
"

HTTP/1.1 100 Continue
Date: Mon, 27 Mar 2017 17:15:26 GMT
Server: CAStor Cluster/9.2.0
Content-Length: 0

HTTP/1.1 201 Created
Location:
http://192.168.1.12:80/surveillance/2017/03/22/20170311-972-9928817883.m
p4 \
 ?domain=example.com
Location:
http://192.168.1.13:80/surveillance/2017/03/22/20170311-972-9928817883.m
p4 \
 ?domain=example.com
Volume: 8aff01dbe86d6ff1f27b5872bfc8e840
Volume: cef223aa1bfc13e356203fdede8489e4
Manifest: ec
Last-Modified: Mon, 27 Mar 2017 17:15:25 GMT
Castor-System-Encoding: zfec 1.4(2, 1, 524288, 200000000)
Castor-System-Version: 1490634925.750
Etag: "c04b7eac90a3f22292581080c32fdd07"
Replica-Count: 2
Date: Mon, 27 Mar 2017 17:17:16 GMT
Server: CAStor Cluster/9.2.0
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400
<html><body>New stream created</body></html>
```

To verify that the video is successfully stored, use a HEAD command:

```
curl --head \
 --location-trusted
"http://example.com/surveillance/2017/03/22/20170311-972-9928817883.mp4"

HTTP/1.1 301 Moved Permanently
Date: Mon, 27 Mar 2017 17:22:50 GMT
Server: CAStor Cluster/9.2.0
Location:
http://192.168.1.12:80/surveillance/2017/03/22/20170311-972-9928817883.m
p4
 ?domain=example.com&auth=2db96e4590e029966aecfd0dd96da7e9
Content-Length: 0
Keep-Alive: timeout=14400

HTTP/1.1 200 OK
Castor-System-CID: fd20ce977b35d0509205b27977d697d3
Castor-System-Cluster: example.com
Castor-System-Created: Mon, 27 Mar 2017 17:15:25 GMT
Castor-System-Name: 2017/03/22/20170311-972-9928817883.mp4
Castor-System-Version: 1490634925.750
Content-Disposition: inline
Content-Type: video/mp4
Last-Modified: Mon, 27 Mar 2017 17:15:25 GMT
x-example-meta-Building: Annex 2
x-example-meta-CameraModel: SWDSK-850004A-US
x-example-meta-End-Time: 2017-03-11T13:00:00.421Z
x-example-meta-Location: 972
x-example-meta-Start-Time: 2017-03-11T12:00:01.678Z
Manifest: ec
Content-Length: 1500964975
Etag: "c04b7eac90a3f22292581080c32fdd07"
Castor-System-Path:
/example.com/surveillance/2017/03/22/20170311-972-9928817883.mp4
Castor-System-Domain: example.com
Volume: 8aff01dbe86d6ff1f27b5872bfc8e840
Date: Mon, 27 Mar 2017 17:22:50 GMT
Server: CAStor Cluster/9.2.0
Keep-Alive: timeout=14400
```

The custom metadata is what makes it possible and practical to identify video of interest. Suppose that an incident occurred in the Annex 2 building; to find surveillance video that might be relevant to the investigation, search the survei llance bucket for video taken at Annex 2 during that time span:

```
curl -i --location-trusted "http://192.168.1.11/surveillance\
 ?domain=example.com\
 &format=json&fields=all\
 &content-type=video/mp4\
 &x-example-meta-Building=Annex%202\
 &x-example-meta-Start-Time:date=<2017-03-11T12:17:23Z\
 &x-example-meta-End-Time:date=>2017-03-11T12:17:23Z"

HTTP/1.1 200 OK
Castor-System-Alias: fd20ce977b35d0509205b27977d697d3
Castor-System-CID: 72203a85b0f9d7a64a7625c114f8a886
Castor-System-Cluster: example.com
Castor-System-Created: Mon, 27 Mar 2017 16:37:38 GMT
Castor-System-Name: surveillance
Castor-System-Version: 1490632658.361
X-Timestamp: Mon, 27 Mar 2017 16:37:38 GMT
Last-Modified: Mon, 27 Mar 2017 17:26:00 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 1
Castor-System-Object-Count: 1
Date: Mon, 27 Mar 2017 17:26:00 GMT
Server: CAStor Cluster/9.2.0
Keep-Alive: timeout=14400
[{
 "sizewithreps": 2251447463,
 "contextid": "fd20ce977b35d0509205b27977d697d3",
 "content_type": "video/mp4",
 "name": "2017/03/22/20170311-972-9928817883.mp4",
 "x_example_meta_end_time:date": "2017-03-11T13:00:00.421Z",
 "@timestamp": 1490635036512,
 "x_example_meta_building": "Annex 2",
 "x_example_meta_location:date": 972000,
 "x_example_meta_location": "972",
 "x_example_meta_cameramodel": "SWDSK-850004A-US",
 "domainid": "72203a85b0f9d7a64a7625c114f8a886",
 "x_example_ meta_start_time:date": "2017-03-11T12:00:01.678Z",
 "hash": "c04b7eac90a3f22292581080c32fdd07",
 "timestamp": 1490635036512,
 "x_example_meta_location:double": 972,
 "last_modified": "2017-03-27T17:15:25.748400Z",
 "bytes": 1500964975,
 "content_disposition": "inline",
"x_example_meta_location:long": 972,
  "x_example_me ta_end_time": "2017-03-11T13:00:00.421Z",
  "x_example_meta_start_time": "2017-03-11T12:00:01.678Z"
}]
```

The search correctly found a video of interest in the surveillance bucket and returned the object: `2017/03/22/20170311-972-9928817883.mp4`

Custom Metadata Typing

- Data Types for Custom Headers
- Querying Custom Metadata by Type
- Upgrading an Existing Index

By default, Swarm indexes custom metadata as strings. However, if your data follows standards for numeric, time, or geospatial formats, Swarm will type your custom metadata, so that you can do query operations and manage the data by type. Swarm applies typing to Custom Metadata Headers on all context objects (domains, buckets) and content objects (named, alias, immutable, untenanted). (v9.2)

> Tip
> If you have string data that might match the pattern for a number or date, you can prevent the additional typing by including a non-digit, non-"e" character (such as ~) in the string.

Data Types for Custom Headers

On POST and COPY, these are the data types that Swarm will parse and recognize in your custom headers:

- String (default)
- Numeric (double and long)
- Date (ISO 8601)
- Geo-point (latitude, longitude)

Swarm Search will index on these types so that you can query against these types in your custom metadata.

> Important
> If you query directly to Elasticsearch rather than through Swarm, be sure to convert dashes (hyphens) to underscores, which is how the headers are indexed for Elasticsearch.

Type	Example Header	Example Search	Notes
String	"x-foo-meta: ASCII string."	&x-foo-meta=ASCII%20string	The default datatype. See the Elasticsearch documentation on String data.
Double	"x-foo-meta: 34567.123"	&x-foo-meta:double=34567.123	See the Elasticsearch documentation on Numeric data.

Long	"x-foo-meta: 93e27"	&x-foo-meta:long= 93e27	To prevent overflow, Swarm indexes numeric values only up to 64 bits of precision. See the Elasticsearch documentation on Numeric data.
Geo-point	"x-foo-meta: 11, -33" "x-foo-meta: (11, -33)" "x-foo-meta: 11.22, -33.44"	&x-foo-meta:geo= 11,-33	A 2-tuple of two numeric values. The first value is `lat` (latitude); the second value is `lon` (longitude). For search operations, evaluations are performed element by element. See the Elasticsearch documentation on Geo-point data.
Date	"x-foo-meta: 1420070400001" (milliseconds) "x-foo-meta: 208704067" (seconds) "x-foo-meta: 2016-12-11" "x-foo-meta: 2016-12-11T12:45:30" "x-foo-meta: 2016-12-11T12:45:30.678Z" "x-foo-meta: 2016-12-11T12:45:30.678-01" "x-foo-meta: 2016-12-11T12:45+01:23"	&x-foo-meta:date= 1420070400001	Counts of time since the epoch can be numbers up to 13 digits: ● Digits between 11 and 13 are read as a date in milliseconds-since-the-epoch ● Digits fewer than 11 are read as a date in seconds-since-the-epoch For dates, the time and zone are optional. An offset from UTC (Zulu) can be appended to the time as ±[hh]:[mm], ±[hh][mm], or ±[hh]. See the ISO 8601 Standard and the Elasticsearch documentation on Date data.

Note

Every custom header value is indexed as String; if its format happens to also match one or more of the other data types, Swarm indexes each of those in addition to String:

```
{...,
  "x-foo-meta" : "34684030120",
  "x-foo-meta:long" : 34684030120,
  "x-foo-meta:double" : 34684030120.0,
  "x-foo-meta:date" : 34684030120,
...}
```

Querying Custom Metadata by Type

When you make SCSP queries to Swarm, you can query for custom metadata that is specifically typed by using the additional fields:

- `&x-foo-meta` will perform string-based queries
- `&x-foo-meta:long` will perform integer/long typed queries
- `&x-foo-meta:double` will perform double typed queries
- `&x-foo-meta:date` will perform date-typed queries
- `&x-foo-meta:geo` will perform geo-point-typed queries

Here is an example of a query that filters between a range of timestamps, stored in custom headers:

```
curl -i --location-trusted
"http://192.168.1.11/surveillance?domain=example.com&format=json&fields=
all\
  &content-type=video/mp4\
  &x_example_meta_Building=Annex%202\
  &x_example_meta_Start_Time:date=>2017-03-11T12:00Z\
  &x_example_meta_End_Time:date=<2017-03-11T12:30Z"
```

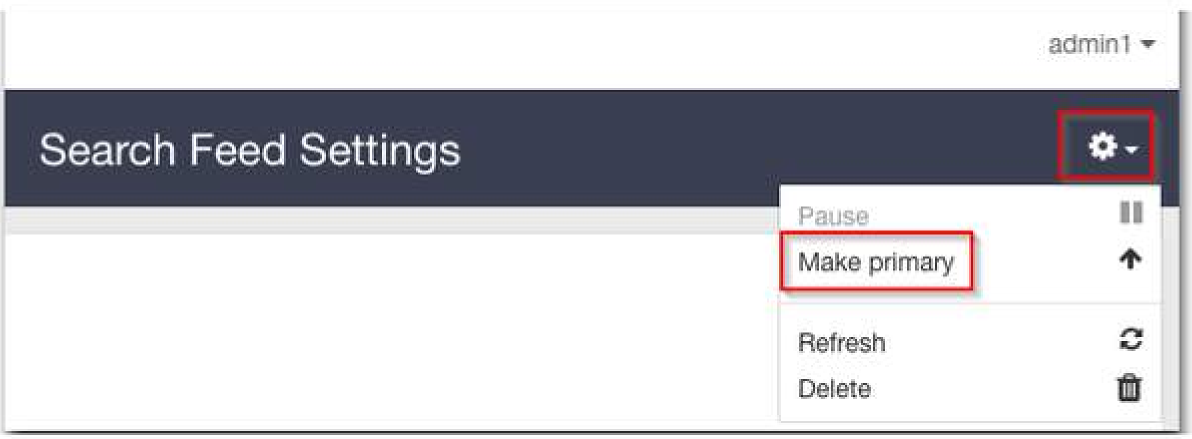
Upgrading an Existing Index

If your existing custom metadata is currently only stored as strings and you want to take advantage of typing across all of your existing custom metadata, you will need to rebuild the index.

When the underlying schema for Swarm Search changes, new feeds are required to generate index data in the new format. Swarm Storage lets you create more than one Search feed so that you can transition from using one feed to another without disruption. During the transition, continue using the primary feed for queries; the second feed is incomplete until it fully clears its backlog. When the second feed is caught up, transition to it (marking it as primary) as soon as reasonable for your operations.

> Important
> When you verify that the new primary feed target is working, delete the original feed. Having two feeds is for temporary use only because every feed incurs cluster activity, even when paused.

1. In the Swarm UI, create a new search feed. Do not select Make primary.
2. Wait until the new feed has completed indexing the cluster, when the feed shows 0 "pending evaluation".
3. When the new feed is ready, make it the primary feed. In the Swarm UI, go to Cluster > Feeds, open the new Search feed, and select Make primary from the drop-down menu.



4. Operate with both feeds for several days. If there is a problem, you can restore the old feed to be primary during troubleshooting.
5. After this confirmation period, delete the old feed. In the Swarm UI, go to Cluster > Feeds, open the old Search feed, and select Delete from the drop-down menu.
6. If desired, delete the old index to reclaim that space.

Encoding Non-ASCII Characters in Metadata

As of version 8.0, Swarm fully supports non-ASCII characters in HTTP headers, which improves metadata querying with Elasticsearch.

How Swarm Handles Non-ASCII

The presence of non-ASCII characters in your object metadata requires extra processing by Swarm. If you encode these characters correctly, Swarm can support these encodings with full metadata searching. Here is a summary of how Swarm handles these characters:

- Swarm tolerates validation failures. Swarm tolerates validation failures and stores header values that are left unencoded, so it does not disturb existing objects whose stored headers might fail decoding under the new header rules. Swarm does not reject any object based on an inability to decode encoded words in a header.

Tip

> Because Swarm does not reject objects with invalid encodings, you are free to adopt new encodings before they are supported by Swarm.

- Swarm stores and returns header fields as-is. Swarm allows all string-typed headers to have multiple lines as well as encoded words. Swarm stores the header value as-is with the object metadata. Only when Swarm needs to use that value (such as for metadata indexing) does it decode the value. Swarm decodes header fields into Unicode and then operates on the decoded values. The original encoded persistent headers, however, remain safely stored with the object and are returned when you perform HEAD or GET operations against the object.

  > Exception
  >
  > You may see different line breaks in multiple-line headers, since Swarm does not store the actual line breaks.

- Metadata goes to Elasticsearch as Unicode. Swarm sends metadata to Elasticsearch as document attributes through the Elasticsearch API. Since the document is passed in the body of the HTTP message as JSON, Swarm must send all non-ASCII characters as Unicode escapes (such as "caf\u00e9" for café). In order to do that, Swarm decodes the metadata using the algorithms specified in RFC 2047.
- Swarm returns Unicode. Swarm returns all query results in a consistent Unicode form, rather than a combination of encoded and non-encoded fields. Swarm query results are returned in the response body, usually as JSON. The JSON results encode non-ASCII characters as Unicode escapes (such as "`caf\u00e9`" for `café`).
- Swarm assumes ISO-8859-1 encoding. Swarm assumes your encoding to be ISO-8859-1, so it treats all octets in request headers as ISO-8859-1 characters. If you have concerns about the treatment of non-ASCII characters, your applications can explicitly encode header values as 'encoded-words' with the ISO-8859-1 character encoding. Swarm decodes any encoded words in headers that specify the ISO-8859-1 encoding.
- Swarm encodes other character sets. To comply with HTTP/1.1 specifications, Swarm headers encode their field content according to the guidance of RFC 2978, which means that characters in sets other than ISO-8859-1 are encoded.

  > Note
  >
  > Decoding affects performance, but the impact is minimal for fields with no special encodings; therefore, you should not see performance impacts unless you store large volumes of non-ASCII metadata in a cluster that enabled full metadata searching.

How to Encode Non-ASCII Characters

Suppose you need to send Swarm a header string with a non-ASCII character, such as `café`. If you do not encode it in ISO-8859-1, you must to encode it as UTF-8, with special treatment:

- Whole-word encoding: `=?UTF-8?Q?caf=C3=A9?=`
- Partial-word encoding: `caf=?UTF-8?Q?=C3=A9?=`

Because there are several ways to encode the same string, Swarm must be able to decode all variations before sending the metadata to Elasticsearch.

For example, you could encode the string "`café red white and blue café blue white red café brown orange`" like this:

```
X-Alt-Meta-Name: =?UTF-8?Q?caf=C3=A9?= red white and blue...
 =?UTF-8?Q?=20caf=C3=A9=20blue=20white=20red=20caf=C3=A9=20brown=20orang
e?=
```

Note
To encode embedded spaces, use ‘`=20`’ or underscores (_).

Best practice
Although Swarm does not force you to, comply with the RFC2047 limits:

- 75 characters per encoded word
- 76 characters per each line with an encoded word

Examples of Decoding

This is how Swarm decodes the following header values:

ASCII	`"alpha beta gamma"`	`'alpha beta gamma'`
UTF-8	`"=?utf-8?q?caf=C3=A9?="`	`u'caf\xe9'`
UTF-8 Base64	`"=?utf-8?b?Y2Fmw6k=?="`	`u'caf\xe9'`
Complex UTF-8	`"=?utf-8?q?caf=c3=a9?= aaa =?utf-8?q?caf=c3=a9?= bbb =?utf-8?q?caf=c3=a9?="`	`u'caf\xe9 aaa caf\xe9 bbb caf\xe9'`
iso-8859-1	`"=?iso-8859-1?q?caf=E9?="`	`u'caf\xe9’`

Valid header values

`"alpha beta gamma"`	Pure ASCII
`"=?utf-8?q?caf=C3=A9?="`	UTF-8
`"=?utf-8?b?Y2Fmw6k=?="`	UTF-8 Base64 encoded

Valid but malformed header values

While valid, incompletely formatted encodings are not decoded because Swarm does not recognize them as having been encoded. That is, when the encoding format is malformed, Swarm treats the content as valid content that was not encoded.

`"=?utf-8?q?caf=C3=A9"`	UTF-8 without the expected suffix
`"=?utf-8?q?caf=C3=A9="`	UTF-8 with a partial suffix
`"utf-8?q?caf=C3=A9?="`	UTF-8 without the expected prefix
`"?utf-8?q?caf=C3=A9?="`	UTF-8 with a partial prefix

Invalid header values

`"=?utf-8?j?caf=C3=A9?="`	UTF-8 with an invalid coding indicator (not “Q” or “B”)
`"=?utf-8?qcaf=C3=A9?="`	UTF-8 missing an internal “?” separator character
`"=?utf-9?q?caf=C3=A9?="`	Invalid or unknown character encoding
`"=?utf-8?q?caf=C3=FF?="`	Invalid or unknown character
`"=?utf-8?b?Y2-mw6k=?="`	Base64 with invalid characters
`"=?utf-8?b?Y2Fmw6k?="`	Base64 with invalid padding

Decoding Limitations

Be aware that Swarm does not do the following:

Feature	Swarm Behavior	Workaround	Example

Disable decoding	Swarm has no configuration parameter to disable decoding of headers. Decoding rules are applied to all header values.	If you need to send a header value that looks like an encoded string but is meant to be taken literally, encode the content itself by encoding as ISO8859-1 with the ? and = replaced by octet values.	To have the header value passed as-is to Elasticsearch (instead of being decoded), you could replace the encoding like this:
			`"=?UTF-8?Q?caf=C3=A9?="`
			`"=?ISO-8859-1?Q?=3D=3FUTF-8?Q=3Fcaf=3DC3=3DA9=3F=3D?="`

Unicode normalizing	Swarm does not perform Unicode normalization.	If you need various encodings of a word such as "`café`" to match, be sure to standardize how you encode such words.	Valid variants for encoding diacritics in UTF-8:
			`=?UTF-8?Q?caf=C3=A9?=`
			`=?UTF-8?Q?caf=65=CC=81?=`

Unicode case folding	Swarm performs no Unicode case folding.	None. For case-insensitive operations, Swarm always converts uppercase to lowercase, including for non-ASCII characters.	In ASCII, uppercasing a character and then lowercasing it always results in the same character. That is not always the case for Unicode escapes.

Troubleshooting Decoding

If Swarm does not decode a header as expected, review these possible reasons why Swarm found the encoding incomplete or invalid:

Problems in encoded word structure

These examples have validation issues in the structure of the encoded-word framework, such as:

- an incorrect starting or ending sequence
- an issue in the "?" separators between the character or Q/B encoding

Swarm passes these types of strings as unencoded text.

Example	Error
`'=?utf-8?Q?_brown=20=20and_blue?'`	Missing the closing '='. Per RFC 2047, encoded word must start with "=?" and end with "?=".
`'=?utf-8Q?_brown=20=20and_blue?='`	Missing the "?" between the "utf-8" and "Q" characters.
`'=?utf-8?J?_brown=20=20and_blue?='`	J encoding is invalid.
`'=?utf-8??Q?_brown=20=20and_blue?='`	Extra "?" before "Q".

Unknown or unreadable encodings

When Swarm encounters an encoded word with an unknown encoding or a valid encoding with any other problem, such as an invalid octet, it passes it through as-is:

Example	Error
`'=?utf-9?Q?_brown=20=20and_blue?='`	utf-9 is not a known encoding.
`'=?utf-8?Q?_brown=20=FFand_blue?='`	0xFF is not a valid octet in utf-8.

Problems with Base64 encoding

If either validation fails in the Base64 encoding, Swarm passes through the original header as-is.

- Characters: Base64-encoded words include only the characters A-F, a-f, +, and /; all other characters are invalid. If any invalid characters are present, Swarm treats the entire encoded word as invalid.
- Padding: Base64 encodings include groups of 4-character sequences. Base64 encodings have trailing padding (with “=”) to maintain the string as a multiple of four characters. Swarm treats any Base64 encodings that lack the trailing padding as invalid.

Problems with one of several encoded words

HTTP header content can contain more than one encoded word, but Swarm does not partially decode headers. If any encoded word in a header is invalid, the entire header is passed through unencoded.

However, if a header includes both complete encoding and incomplete encoding (text that looks like an encoded word missing either the leading “=?” prefix or ending “?=” suffix), Swarm will ignore the incomplete encoding (treating it like a valid non-encoded word) and decode the complete word.

Metadata Annotation

In addition to updating object metadata directly (via COPY), you can effectively append additional metadata to existing objects without altering the original. This gives you a way to extend the metadata of immutable objects, including historical versions, because each object's create date, original metadata, and version sequence remain undisturbed. Annotations give you an additional method for finding and managing your objects, such as storing S3 object-level ACLs for the Gateway to enforce.

> Important
> Once you make use of this feature, you cannot downgrade to an older version of Swarm.

Benefits - Keeping metadata annotation separate from the object itself gives several advantages:

- You can add helpful metadata without changing the object’s create date, original metadata, and version sequence.
- You retrieve objects as originally written, so applications can distinguish between what was original and what was added later.
- You may annotate immutable objects.

- You may annotate historical versions of objects, independent of the current version of the object. This is keenly important when the metadata is derived from analysis performed on the data, which changes from version to version, or when capturing information about specific versions.

> Note
> This lightweight implementation of annotation does not rely on the annotator and the target object interacting, and the objects do not operate as a pair. For example, there is no single request that returns both objects' headers, and there is no method to merge and resolve conflicts between them.

Annotation Cleanup

There are two key features of this annotation method: (1) validation that target objects exist before annotations are written, and (2) the Health Processor's automated tracking and cleanup of annotation objects after the target object is removed. A target object that you annotate might be removed from Swarm in one of several ways:

- SCSP Delete
- SCSP Write (invalidating the old version)
- Lifepoint Delete
- Recursive delete of a parent context (domain or bucket)

> Note
> When the Health Processor purges an annotation during garbage collection, it logs a “DECORATION DELETE” AUDIT-level message. Annotation objects "decorate" a targeted content object.

Regardless of the type of Swarm object you annotate (named, alias, immutable, historical version) and its protection type (replicated or erasure-coded), metadata annotation behaves largely the same way:

- If you create an annotation and later delete its target object, Swarm will delete the orphaned annotation during garbage collection.
- If you create an annotation and later delete the annotation, the target object is completely unaffected.
  - For named objects only, Swarm replaces the annotation object with a delete marker.
- If you delete a domain or a bucket that contains both the original object and its annotation, Swarm will delete both recursively.
- If you update a versioned object, you can create separate annotations for any of the historical versions; when you delete a version, Swarm will delete its orphaned annotation during garbage collection.
- If you create and later delete an annotation on a versioned object, it differs by its position in the version chain:
  - Historical versions: Swarm simply removes the annotation.
  - Current versions: Swarm replaces the annotation object with a delete marker.

Creating Annotations

Metadata annotation makes use of a persisted header, `Castor-System-Decorates`, which is the ETag of the target object that the annotation object is extending (decorating). Simply, if this header is present, then this is an annotation object, subject to special Health Processor management. The header is valid for all Swarm object types (immutable, alias, and named), but not for context objects (domains and buckets). Both the annotator (decorator) and annotated target object may be versioned.

Tip

Although it is common to create annotations as metadata-only (`Content-Length: 0`) objects, annotations are complete objects in their own right, which means that you may include data as part of the annotation.

When you write a new annotation object, you create a new object that points to the ETag of its target and includes the custom metadata to be added, such as GPS coordinates extracted from an existing, uploaded photo:

Extending metadata with post-processed data

```
Content-Length: 0
Castor-System-Decorates: 9282727ffcca3a09e0843281aafc13af
X-GPS-Meta-Longitude: 36; 16; 48.36000000000589
X-GPS-Meta-Latitude: 115; 10; 20.79299999981990
```

Note that the ETag has no quotes, even though returned ETags are quoted. The ETag must be of a content-containing object, not a delete marker.

Alias objects

You must use the current ETag of an alias object, not its permanent UUID, for creating annotations. If you use the UUID, Swarm returns a 400 - Bad Request error.

Searching for Annotations

In the annotation (decorator) object's Elasticsearch record, the `Castor-System-Decorates` header value is indexed under the key decorates, and the Elasticsearch configuration templates include the decorates field. Most Swarm queries will return this value, if present, as part of the results.

Query argument - You can use a "`decorates=<uuid>`" query argument in Swarm listing queries to find annotation objects for a given ETag (or earlier query result "hash").

> See Listing Operations.

Sample Scenario for Annotations

Suppose that a company needs to store its surveillance videos as immutable objects (as protection from tampering) in the domain "`swarm.example.com`". To add a video, use the normal POST, adding the Content-Type of the video and custom metadata for the video's duration, camera location, and camera model:

```
curl -i --location-trusted -X POST --post301 \
--data-binary @20170311-972-9928817883.mp4 \
-H "Expect: 100-continue" \
-H "x-example-meta-Start-Time: 2017-03-11T12:00:01.678Z" \
-H "x-example-meta-End-Time: 2017-03-11T13:00:00.421Z" \
-H "x-example-meta-Building: Annex 2" \
-H "x-example-meta-Location: 972" \
-H "x-example-meta-CameraModel: SWDSK-850004A-US" \
-H "Content-Type: video/mp4" \
-H "Content-Disposition: inline" \
"http://swarm.example.com/"

HTTP/1.1 201 Created
Location:
http://192.168.1.11:80/e970b3280d5501571c8c6fe9d6838557?domain=swarm.exa
mple.com
Location:
http://192.168.1.12:80/e970b3280d5501571c8c6fe9d6838557?domain=swarm.exa
mple.com
Volume: b3381183a1cfc620d960db3eae1d086d
Volume: 604a44d1a351045553b5481391af0810
Manifest: ec
Content-UUID: e970b3280d5501571c8c6fe9d6838557
Last-Modified: Tue, 28 Mar 2017 19:19:48 GMT
Castor-System-Encoding: zfec 1.4(2, 1, 524288, 200000000)
Castor-System-Version: 1490728788.934
Etag: "681b2470307b9260fb83542903e51828"
Replica-Count: 2
Date: Tue, 28 Mar 2017 19:22:19 GMT
Server: CAStor Cluster/9.2.0
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400
<html><body>New stream created</body></html>
```

To verify that the video is successfully stored, use a HEAD command:

```
curl --head --location-trusted
"http://swarm.example.com/e970b3280d5501571c8c6fe9d6838557"

HTTP/1.1 200 OK
Castor-System-CID: 7e7fd5d747d244726af93c726672408b
Castor-System-Cluster: swarm.example.com
Castor-System-Created: Tue, 28 Mar 2017 19:19:48 GMT
Content-Disposition: inline
Content-Type: video/mp4
Last-Modified: Tue, 28 Mar 2017 19:19:48 GMT
x-example-meta-Building: Annex 2
x-example-meta-CameraModel: SWDSK-850004A-US
x-example-meta-End-Time: 2017-03-11T13:00:00.421Z
x-example-meta-Location: 972
x-example-meta-Start-Time: 2017-03-11T12:00:01.678Z
Manifest: ec
Content-Length: 1500964975
Etag: "681b2470307b9260fb83542903e51828"
Castor-System-Domain: swarm.example.com
Volume: b3381183a1cfc620d960db3eae1d086d
Date: Tue, 28 Mar 2017 19:24:25 GMT
Server: CAStor Cluster/9.2.0
Keep-Alive: timeout=14400
```

The custom metadata is what makes it possible and practical to identify video of interest. Suppose that an incident occurred in the Annex 2 building; to find surveillance video that might be relevant to the investigation, search for immutable video taken at Annex 2 during that time span:

```
curl -i --location-trusted
"http://swarm.example.com/?domain=swarm.example.com&format=json&fields=a
ll\
&stype=immutable\
&content-type=video/mp4\
&x-example-meta-Building=Annex%202\
&x-example-meta-Start-Time:date=<2017-03-11T12:17:23Z\
&x-example-meta-End-Time:date=>2017-03-11T12:17:23Z"

HTTP/1.1 200 OK
Castor-System-Alias: 7e7fd5d747d244726af93c726672408b
Castor-System-CID: ffffffffffffffffffffffffffffffff
Castor-System-Cluster: swarm.example.com
```

```
Castor-System-Created: Tue, 28 Mar 2017 19:19:29 GMT
Castor-System-Name: swarm.example.com
Castor-System-Owner: @CAStor administrator
Castor-System-Version: 1490728769.536
X-Timestamp: Tue, 28 Mar 2017 19:19:29 GMT
Last-Modified: Tue, 28 Mar 2017 19:26:30 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 1
Castor-System-Object-Count: 1
Date: Tue, 28 Mar 2017 19:26:30 GMT
Server: CAStor Cluster/9.2.0
Keep-Alive: timeout=14400
[{
   "contextid":"7e7fd5d747d244726af93c726672408b",
   "x_example_meta_start_time":"2017-03-11T12:00:01.678Z",
   "x_example_meta_end_time:date":"2017-03-11T13:00:00.421Z",
   "@timestamp":1490728939869,
   "domainid":"7e7fd5d747d244726af93c726672408b",
   "last_modified":"2017-03-28T19:19:48.932400Z",
   "bytes":1500964975,
   "hash":"681b2470307b9260fb83542903e51828",
   "x_example_meta_location:double":972.0,
   "content_disposition":"inline",
   "sizewithreps":2251447463,
   "content_type":"video/mp4",
   "timestamp":1490728939869,
   "x_example_meta_location:long":972,
   "x_example_meta_location:date":972000,
   "x_example_meta_end_time":"2017-03-11T13:00:00.421Z",
   "name":"e970b3280d5501571c8c6fe9d6838557",
   "castor_stream_type":"immutable",
   "x_example_meta_building":"Annex 2",
   "x_example_meta_location":"972",
   "x_example_meta_cameramodel":"SWDSK-850004A-US",
   "x_example_meta_start_time:date":"2017-03-11T12:00:01.678Z"
}]
```

The search correctly finds a video of interest: `e970b3280d5501571c8c6fe9d6838557`

Adding Metadata Annotation

With the video stored securely, suppose the organization also needs to run an application to perform facial recognition on the video. When the application is run, it will generate data of its own, including both information on the algorithm/settings and the detailed results. The original video object must remain read-only to serve as evidence, so the derived data and metadata must be stored in a way that associates it with the original object without altering it.

The solution is to annotate the video with a decoration object (which can be named or unnamed) to associate the results with the original video.

Important

The `Castor-System-Decorates` header always refers to the ETag of the original video, not its GUID; this is a precaution against the annotation becoming orphaned if the target object is mutable.

```
curl -i -X POST --post301 --location-trusted -d @results \
-H "Castor-System-Decorates: 681b2470307b9260fb83542903e51828" \
-H "x-VideoAnalysis-meta-Algorithm: facial-recognition" \
-H "x-VideoAnalysis-meta-Version: 8.7" \
-H "Content-Type: application/vnd.analysis.facerec" \
"http://swarm.example.com"

HTTP/1.1 201 Created
Location:
http://192.168.1.13:80/0cb2d9e90a3341b10bc9dba27f27259c?domain=swarm.exa
mple.com
Location:
http://192.168.1.12:80/0cb2d9e90a3341b10bc9dba27f27259c?domain=swarm.exa
mple.com
Volume: 6bd38289c2a8fb314caf902d9811fb87
Volume: 604a44d1a351045553b5481391af0810
Manifest: ec
Content-UUID: 0cb2d9e90a3341b10bc9dba27f27259c
Last-Modified: Tue, 28 Mar 2017 20:26:08 GMT
Castor-System-Encoding: zfec 1.4(2, 1, 524288, 200000000)
Castor-System-Version: 1490732768.888
Etag: "867c10c9e6649313a3a5eed2cc76f307"
Replica-Count: 2
Date: Tue, 28 Mar 2017 20:26:12 GMT
Server: CAStor Cluster/9.2.0
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400
<html><body>New stream created</body></html>
```

Tip

You can create annotations as part of the intake process that stores the original objects in Swarm.

To find any annotations producing facial recognition on the original object, search for objects that decorate the video and also qualify the search to look for only facial recognition results:

```
curl -i --location-trusted
"http://swarm.example.com/?domain=swarm.example.com&format=json&stype=im
mutable&fields=all\
&decorates=681b2470307b9260fb83542903e51828\
&x_videoanalysis_meta_algorithm=facial%20recognition"

HTTP/1.1 200 OK
Castor-System-Alias: 7e7fd5d747d244726af93c726672408b
Castor-System-CID: ffffffffffffffffffffffffffffffff
Castor-System-Cluster: swarm.example.com
Castor-System-Created: Tue, 28 Mar 2017 19:19:29 GMT
Castor-System-Name: swarm.example.com
Castor-System-Owner: @CAStor administrator
Castor-System-Version: 1490728769.536
X-Timestamp: Tue, 28 Mar 2017 19:19:29 GMT
Last-Modified: Tue, 28 Mar 2017 20:36:40 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 1
Castor-System-Object-Count: 1
Date: Tue, 28 Mar 2017 20:36:40 GMT
Server: CAStor Cluster/9.2.0
Keep-Alive: timeout=14400
[{
   "sizewithreps":11684987,
   "contextid":"7e7fd5d747d244726af93c726672408b",
   "content_type":"application/vnd.analysis.facerec",
   "name":"0cb2d9e90a3341b10bc9dba27f27259c",
   "castor_stream_type":"immutable",
   "timestamp":1490732772033,
   "@timestamp":1490732772033,
   "domainid":"7e7fd5d747d244726af93c726672408b",
   "decorates":"681b2470307b9260fb83542903e51828",
   "x_videoanalysis_meta_algorithm":"facial-recognition",
   "x_videoanalysis_meta_version:long":9,
   "x_videoanalysis_meta_version:date":8700,
   "x_videoanalysis_meta_version":"8.7",
   "hash":"867c10c9e6649313a3a5eed2cc76f307",
   "last_modified":"2017-03-28T20:26:08.888400Z",
   "x_videoanalysis_meta_version:double":8.7,
   "bytes":7789991
}]
```

The search correctly finds an annotation: `0cb2d9e90a3341b10bc9dba2`

> Note
> Elasticsearch is a NoSQL (non-relational) database that does not support joins directly, so you cannot combine queries for a primary object and an annotation object.

Content Integrity

Content integrity refers to the accuracy and consistency (validity) of content over its lifecycle in Swarm storage. Integrity can be lost at various levels:

- Human error or tampering
- Transfer errors, including unintended alterations or data compromise going to or from storage
- Cyber threats (bugs, viruses/malware, hacking)
- Compromised hardware, such as a device or disk crash
- Physical compromise to devices

Best practices for content integrity include multiple approaches:

- Input validation, to preclude the entering of invalid data
- Error detection/data validation, to identify errors in data transmission
- Security measures, such as access control and data encryption

- Content Integrity Assurance
- Content-MD5 Checksums
- Caching Metadata Headers

Content Integrity Assurance

- Integrity seals
- Validating reads
- Application-initiated hash upgrading

Swarm provides methods for allowing applications to obtain and validate integrity guarantees on the stored data. In this context, integrity is an independently verifiable guarantee that the data returned for a given name or UUID is exactly the same data that was stored using that name or UUID, perhaps many months or years in the past. This is done by has hing the data using a cryptographic hash algorithm.

Content metadata is not included in the hash. If the application stores the name or UUID and its associated hash value, these can be used later to verify the content has not changed, either through accidental or malicious means.

Integrity seals

An integral seal is a URL containing the object name or UUID, its hash value, and the type of hash algorithm that was used for the computations. An application can request an integrity seal when it performs a WRITE by including a `hashtype` query string.

Example of a hashtype request

```
POST http://company.cluster.com/?hashtype=md5 HTTP/1.1
```

These are the current allowable hash types:

- md5
- sha1
- sha256
- sha384
- sha512

After creating the object and assigning a name or UUID, Swarm replies with a 201 (Created) response that includes a location header with a URL that can later be used to retrieve the data.
In addition to the host and name or UUID, the URL includes the hash type and value computed from the content object. This URL, including the triple name or UUID, hash type, and hash, is known as the content object integrity seal.

Example of a complete integrity seal embedded in a Location header

```
Location: http://129.69.251.143/41A140B5271DC8D22FF8D027176A0821
 ?hashtype=md5
 &hash=7A25E6067904EAC8002498CF1AE33023
```

Validating reads

An integrity seal can be used in a subsequent READ request to validate the data stored in a storage cluster (any cluster). By supplying the URL returned in the Location header from the WRITE request (perhaps replacing the host address if connected to a different cluster or node), the application can ask Swarm to validate while reading the data.

Example of validation with read

```
GET http://129.69.251.143/41A140B5271DC8D22FF8D027176A0821
 ?hashtype=md5
 &hash=7A25E6067904EAC8002498CF1AE33023 HTTP/1.1
```

When Swarm receives such a READ request, it recomputes the hash of the stored content using the supplied hash type and compares the computed hash with the hash value in the integrity seal.

- Match - If the content was not modified or corrupted in any way, the hashes match. Swarm returns the object with the computed digest as a trailing `Location` header.

- No match - If the two values do not match, Swarm will drop the connection before sending the object content at the end of the request.

Because the hash algorithms are published and well-known, users and third parties can independently validate an object that was stored by Swarm by reading its contents, computing the hash value, and comparing it with the hash value in the seal. By publishing an integrity seal when it is created, you can ensure that the stored content was not modified and it has always been associated with the same UUID.

> Important
>
> Range headers are not compatible with integrity seals. If the seal is incorrect, the connection might be closed prematurely.

Application-initiated hash upgrading

Occasionally, cryptographers and mathematicians may defeat a cryptographic algorithm, making it possible for hackers to generate different content that has exactly the same hash value as previously-stored content. This issue occurred with the md5 and sha1 algorithms, but not the sha256, sha384, or sha512 algorithms.

Unlike other fixed content storage solutions, Swarm allows a user or application to upgrade a hash algorithm for an existing individual integrity seal. This is done by issuing a READ request with the name or UUID, the current hash type and hash, and then specifying a different, presumably stronger, hash type in the newhashtype query parameter.

> Important
>
> Upgrade the hash promptly, before any exploit of the old algorithm becomes well known and available.

Example of hash upgrading

```
GET http://129.69.251.143/41A140B5271DC8D22FF8D027176A0821
 ?hashtype=md5
 &hash=7A25E6067904EAC8002498CF1AE33023
 &newhashtype=sha256 HTTP/1.1
```

This READ request will first validate the given integrity seal, then reseal it by wrapping the content in the new, upgraded hash algorithm – sha256 in the example. If the requested object fails to validate against its integrity seal, Swarm will send a 200 OK response but will drop the connection prior to sending the object content. If the object validates properly, a new integrity seal will be returned with the new hash type and hash value.

Content-MD5 Checksums

- Client-Provided Content-MD5
- Swarm-Provided Content-MD5
- Storing Content-MD5 Headers
- Content-MD5 and Replication
- Content-MD5 and Erasure-Coding

Content-MD5 checksums provide an end-to-end message integrity check of the content (excluding metadata) as it is sent to and returned from Swarm. A proxy or client can check the Content-MD5 header to detect modifications to the entity-body while in transit. Additionally, a client can provide this header to indicate that Swarm should compute and check it as it is storing or returning the object data.

> See SCSP Headers.

Client-Provided Content-MD5

During a POST or PUT, the client can provide the following Content-MD5 header as specified in section 14.15 of the HTTP/1.1 RFC:

```
Content-MD5 = "Content-MD5" ":" md5-digest
```

where md5-digest is the base64 of the 128-bit MD5 digest (see RFC 1864 for more information).

The md5-digest is computed based on the content of the entity body, including any content coding that was applied, but not including any transfer-encoding applied to the message body.

- If this header is present, Swarm computes an MD5 digest during data transfer and then compares the computed digest to the digest provided in the header.
- When completed, the Content-MD5 data is stored with the object and returned with the GET or HEAD request.
- If the hashes do not match, Swarm returns a 400 Bad Request error response, abandons the object, and closes the client connection.

Swarm-Provided Content-MD5

Another way to associate a Content-MD5 value with an object is to have Swarm compute the ContentMD5 for the body data of the request. To do this, include the gencontentmd5 query argument in the request. Swarm will return the Content-MD5 as a header in the 201 Created response. Once computed, the Content-MD5 data is stored with the object and returned as a response header for any subsequent GET or HEAD requests. Note that the gencontentmd5 query argument replaces use of the "Expect: Content-MD5" request header, which is deprecated per RFC 2731. (v9.2)

> Tip
> The Swarm setting scsp.autoContentMD5Computation automates Content-MD5 hashing, which means that you do not need to include the gencontentmd5 query argument or the deprecated Expect: Content-MD5 header on writes (although you may want to supply your own Content-MD5 header for content integrity checking). This setting is ignored wherever it is invalid, such as on a multipart initiate/complete or an EC APPEND. (v9.1)

Ranges - When you include ?gencontentmd5 on a GET request with a Range header, any Content-MD5 header stored with the object will be omitted in the response headers. Instead, a Content-MD5 of the selected range will be returned as a trailing header to the GET request.

> For details about Range headers, see section 14.35 (Range) in the HTTP/1.1 RFC

Storing Content-MD5 Headers

Content-MD5 headers are stored with the object metadata and returned on all subsequent GET or HEAD requests.

- If a Content-MD5 header is included with a GET request, Swarm computes the hash as the bytes are read,

regardless of whether the header was originally stored with the object
- If the computed and provided hashes do not match, the connection is closed before the last bytes are transmitted, which is the standard way to indicate something went wrong with the transfer.

Content-MD5 and Replication

When you provide the gencontentmd5 query argument in a request on a replicated object, the following applies:

- On a write request (POST, PUT, COPY, or APPEND), the Content-MD5 is calculated, stored with the object, and returned as a response header for that write operation.
- The Content-MD5 is always returned for any GET or HEAD request that was written with the gencontentmd5 query argument.
- When you include ?gencontentmd5 on a range read (a GET request with the Range header), Swarm will suppress any stored Content-MD5 from the response headers and instead return a Content-MD5 for the requested range as a trailing header.

Content-MD5 and Erasure-Coding

When you provide the gencontentmd5 query argument in request on an erasure-coded object, the following applies:

- The APPEND operation is no longer supported. If you provide a gencontentmd5 query argument on an APPEND, it returns a 400 Bad Request error response.
- The COPY operation is only supported if you provided a gencontentmd5 query argument on the existing object's write. Otherwise the COPY operation fails.
- For a range read (a GET request with the Range header), Swarm will suppress any stored Content-MD5 from the response headers and instead return a Content-MD5 for the requested range as a trailing header.

Caching Metadata Headers

Caching metadata headers let clients and caching proxies quickly determine if a resource was modified since the last time it was read. With alias objects, caching headers let clients verify that the previous read is the current revision before writing an update to it.

> See SCSP Headers.

HTTP defines several header mechanisms for clients and caching proxies to quickly determine whether a resource was modified since the last time the data was read. In the Swarm context, caching headers make proxies more effective by extending the caching period to its maximum value, essentially telling the proxies that the resource will not change for immutable objects.

To maintain compatibility with a wide variety of browsers and proxies, Swarm implements the caching mechanisms for both HTTP/1.0 and HTTP/1.1.

> Note
> Swarm will return a 412 response if it cannot find the bucket or domain associated with a request. This can be distinguished from a cache response by the lack of the current ETag in the response headers and a response body that denotes that the bucket or domain cannot be located.

HTTP 1.1 Caching Headers

The newer HTTP/1.1 cache coherency mechanism does not use dates or timestamps and thus avoids the granularity and synchronization problems of the HTTP/1.0 headers. Instead, it uses entity tags (or ETags) that can be compared only for exact equality.

In Swarm, ETag values are opaque, variable length, case-sensitive strings that must be enclosed in quotes. Any characters preceding or following the quoted string are ignored. If the header value has no quoted string, the entire header is ignored. The value of each date header adheres to the Full Date specification(RFC 7232), and only dates in that format are recognized by Swarm on incoming requests.

Swarm supports the following HTTP 1.1 caching headers:

- Cache-Control (RFC 7234 5.2)
- ETag (RFC 7231 2.3)
- If-Match (RFC 7232 3.1)
- If-None-Match (RFC 7232 3.2)
- If-Range (RFC 7233 3.2)

Cache-Control

Cache-Control can be used on READ and WRITE requests to determine whether data retrieved from the content cache is acceptable for this request or whether a specific object can ever be stored in the content cache. Swarm supports the C ache-Control: no-cache and Cache-Control: max-age parameters as discussed in RFC 7234 5.2.

Swarm also supports the Cache-Control: no-cache-context extension that instructs Swarm not to use cached contexts. (A context is a container; for example, the context of a named object is a bucket.) Cache-Control: no-cache-context can be used on any SCSP READ or WRITE request to instruct Swarm to ignore the content cache when looks up the bucket and domain for a named object. For example, you can use it in a READ request to prevent Swarm from returning "stale" bucket and domain data from the cache.

> See Use the Content Cache in a Distributed System for when Swarm might return "stale" data.

ETag

Swarm returns the ETag header for all POST, PUT, COPY, APPEND, GET, and HEAD operations. Swarm will use only "strong" ETags (as defined in RFC 7232 2.3) that can be compared only for exact (case-sensitive) equality.

Example of an ETag response header:

```
ETag: "508941dc9b52243f64d964b058354b76"
```

The ETag of an immutable unnamed object never changes during the entire lifecycle of the object, whereas mutable named and unnamed object ETags change each time the object is mutated by a PUT.

> Note
> You cannot perform SCSP operations (Update, Delete, etc.) for an existing object using the ETag.

If-Match

A Swarm client or proxy can include the If-Match header with the PUT, COPY, APPEND, GET, and HEAD methods. The value of the header is either a single quoted string (possibly with some ignored flags outside the quotation marks), a comma-separated list of quoted strings, or a single asterisk. Any additional strings will be ignored.

Below are examples of If-Match request headers:

```
If-Match: "508941dc9b52243f64d964b058354b76"
If-Match: "508941dc9b52243f64d964b058354b76",
"fe3233d3c6881d5e8b654117b829d26c"
If-Match: W/"508941dc9b52243f64d964b058354b76"
If-Match: *
```

If any of the entity tags match the primary UUID of the object that would have been returned in the response to a similar GET request (without the If-Match header) on that resource or if "*" is given, Swarm performs the requested method as if the If-Match header field did not exist. If the request results in anything other than a 2xx status without the If-Match header field, the If-Match header will be ignored.

If none of the entity tags match, Swarm will not perform the requested method, and will instead return a 412 Precondition Failed response with a current ETag header. This behavior is most useful when the client wants to prevent an updating method (such as PUT) from modifying an aliased object that changed since the client last retrieved it.

If Swarm receives a conditional request that includes both a Last-Modified date (for example, in an If-Modified-Since or If-Unmodified-Since header field) and one or more entity tags as cache validators (for example, in an If-Match header field), Swarm will not return a response status of 412 Precondition Failed unless it is consistent with all of the conditional header fields in the request.

If-None-Match

A Swarm client or proxy can include this header with the PUT, COPY, APPEND, GET, and HEAD methods to make it conditional. This feature allows efficient cached information updates with a minimum amount of transaction overhead. The header value is either a single quoted string (possibly with some ignored flags outside the quotation marks), a comma-separated list of quoted strings, or a single asterisk, anything after which will be ignored.

Examples of If-None-Match request headers:

```
If-None-Match: "508941dc9b52243f64d964b058354b76"
If-None-Match: "508941dc9b52243f64d964b058354b76",
"fe3233d3c6881d5e8b654117b829d26c"
If-None-Match: W/"508941dc9b52243f64d964b058354b76"
If-None-Match: */
```

If any of the entity tags match the primary object UUID that would have been returned in the response to a similar GET request (without the If-None-Match header) on that object or if "*" is given and the object does exist, Swarm will not perform the requested method. If the request method was GET or HEAD, Swarm will respond with a 304 Not Modified response, including a current ETag header for the object. For all other request methods, Swarm will respond with a response of 412 Precondition failed with the same current ETag as the GET or HEAD response. If none of the previously recorded and supplied entity tags match, the object was modified. As a result, the requested method will proceed as if the If-None-Match header field did not exist.

If Swarm receives a conditional request that includes both a Last-Modified date (for example, in an If-Modified-Since or If-Unmodified-Since header field) and one or more entity tags (for example, in an If-None-Match header field) as cache validators, Swarm will not return a response status of 304 Not Modified or 412 Precondition failed unless it is consistent with all of the conditional header fields in the request.

If-Range

A Swarm client or proxy can include the If-Range header with a GET request method to obtain an additional specified portion of the object if it has not changed or the entire object if it has changed. The value of the header can be either a single quoted string (possibly with some ignored flags outside the quotation marks) or an HTTP-date string (unquoted).

Examples of If-Range request headers:

```
If-Range: "508941dc9b52243f64d964b058354b76"
If-Range: W/"508941dc9b52243f64d964b058354b76"
If-Range: Tue, 07 Jul 2009 16:25:24 GMT
```

If a client has a partial copy of an object in its cache and wishes to have an up-to-date copy of the entire object in its cache, it could use the Range request-header with a conditional GET using either or both of If-Unmodified-Since and If-Match headers. If the condition fails because an aliased object was updated, the client would have to make a second request to obtain the entire current object. The If-Range header allows a client to "short-circuit" the second request. Informally, its meaning is "if the object is unchanged, send me the part(s) that I am missing; otherwise, send me the entire object."

If the client has no entity tag for an object but has a Last-Modified date, it can use that date in an If-Range header. Swarm can distinguish between a valid HTTP-date and any form of entity-tag by looking for double quotes. The If-Range header should only be used together with a Range header, and will be ignored if the request does not include a Range header.

If the entity tag given in the If-Range header matches the current primary object UUID or the HTTP-date given is not before the Last-Modified date of the object, Swarm will provide the specified sub-range of the object using a 206 Partial content response. If the entity tag does not match, Swarm will return the entire object using a 200 OK response.

HTTP 1.0 Caching Headers

In the first version of HTTP, the cache coherency mechanism used time stamps with one-second granularity to decide if a resource was modified and, therefore, required invalidating the cached copy. In addition to the course time granularity that could mask changes made in the same second (to aliased objects for example), this approach also requires the client and/or proxy clocks to be reasonably well synchronized with the server clocks.

> Warning
>
> Although Swarm supports this coherency method for compatibility reasons, it is not the preferred mechanism because of these issues and is not supported for rapid update use cases. ETag comparisons are recommended for cache coherency on objects that are rapidly updated. The value of each date header adheres to the Full Date Section 3.3.1 of the HTTP/1.1 specification and only dates in that format are recognized by Swarm on incoming requests.

Swarm supports the following HTTP/1.0 caching headers:

- Last-Modified
- If-Modified-Since
- If-Unmodified-Since
- Expires

Last-Modified

Swarm returns the Last-Modified header for all POST, PUT, COPY, APPEND, GET, and HEAD operations. For both ordinary objects and aliased objects, the value of the header will be exactly the same as the Castor-System-Created header.

- For ordinary objects, this is the original object time stamp.
- For aliased objects, this is the server time when the alias was last updated.

> Castor-System-Created deprecated
>
> The Castor-System-Created header is deprecated, replaced with the more standard Last-Modified header. For backward compatibility with previously stored data, Swarm will continue to generate both headers and behave as it does now if it encounters an object with a Castor-System-Created header, but without a Last-Modified header. If a stored object includes both headers, Swarm will use the value of the Last-Modified header. A future release will cease generating the deprecated header for newly-stored content.

If-Modified-Since

A Swarm client or proxy can include the If-Modified-Since header with a GET or HEAD method request. All other methods ignore the header when present in the request. The If-Modified-Since request header field is used with a GET to make it conditional.

> Note
>
> If-Modified-Since is for use with GET and HEAD requests only (not writes). If you specify a date in the future, Swarm will ignore it.

If the requested object was not modified since the time specified in the If-Modified-Since header, an entity will not be returned from the server. Instead, a 304 Not Modified response is returned without any message-body.

> See Section 14.25 in the HTTP 1.1 specification for details.

> Best practice
> If you have mutable objects that are frequently updated, use ETag comparisons, which offer cache coherency on objects that are rapidly updated.

If-Unmodified-Since

A Swarm client or proxy can include this header with a GET, PUT, or DELETE method. All other methods ignore this header when present in the request. The If-Unmodified-Since request header field is used with a method to make it conditional.

- If the requested object was not modified since the time specified in this field, Swarm performs the requested method as if the If-Unmodified-Since header were not present.
- If the requested object was modified since the specified time, Swarm will not perform the requested method, and instead, return a 412 Precondition failed.
- If the specified date is invalid, the header is ignored.

Expires

Swarm returns an Expires header if it is persisted with your content. Swarm does not generate an Expires header.

The Expires header field provides the final date and time when the response is considered stale. A stale cache entry may not normally be returned by a cache (either a proxy cache or a user agent cache) unless it is first validated with Swarm (or with an intermediate cache that has a fresh copy of the object). Since Swarm has no information about when an aliased object might be updated and little information about when an object might be deleted, Swarm does not generate an Expires header for any object. However, Expires will be added to the list of persisted headers so that applications can supply a hint to caching proxies and clients as to when an object might become stale.

Multipart Write

> Note
> Multipart Write was previously referred to as Parallel Write; the functionality is the same.

With Multipart Write, you can upload parts of a large object from multiple clients at the same time. Multipart Write lets your client application split a large file into multiple pieces, transfer the pieces concurrently to Swarm, and then request that Swarm combine the separately uploaded parts together as a single object, thereby minimizing the upload time.

Multipart write requires erasure coding (EC). The health processor (HP) has the ability to consolidate the segments of erasure-coded objects that have sub-optimal segment usage, such as can happen when you do multipart writes of objects using small parts. To enable consolidation, set the configuration setting, ec.segmentConsolidationFrequency, to 5 (recommended), which performs all consolidations over 20 HP cycles, if consolidation is needed. (v9.1)

> Tip
> Every multipart write must be erasure-coded for upload; however, if the uploaded object does not meet the current policy for EC encoding, the HP converts it to a replicated object. To maintain erasure coding for the lifetime of the object, be sure to add a lifepoint to that effect.

To upload a large object in parts using multipart write, you must perform three distinct actions in this order:

1. Initiate a multipart write.
2. Upload or copy the parts.
3. Complete or cancel the procedure.

- Initiating Multipart Write
- Uploading the Parts
- Completing the Multipart Write
- Canceling a Multipart Write
- Validating a Multipart Write
- Multipart Write Example

Initiating Multipart Write

- Multipart writing a new named object
- Multipart writing an immutable object
- Multipart writing an existing object

A multipart, parallel upload is started with an initiate request that can be an SCSP POST, PUT, or APPEND request. If you are appending to an existing object, the object must already be erasure-coded or the request will fail. To convert a replicated object to EC, use a 0-byte APPEND with the query arguments `erasurecoded=yes` and `encoding=<k:p>`.

Query arg - The initiate request must include the `uploads` query argument, and it can specify an immutable, named, or aliased object. The object encoding is determined by the `encoding` query argument; if the argument is missing, the applicable encoding policy stands. Any body text included on the initiate request is ignored and not included in the final object.

Upload ID - When the request is completed, Swarm returns an upload ID that identifies the upload. This ID serves as the identifier for all subsequent operations associated with the upload. The name or resulting Content-UUID value from the initiate request is the object name that must be used for subsequent operations for the same upload ID.

> Important
>
> Do not include a Content-MD5 header in the initiate or complete request. This operation is unsupported and returns a 400 Bad Request error response. To generate a Content-MD5 for an individual part or the completed object, use a separate POST request with the gencontentmd5 query argument.

Multipart writing a new named object

Example of initiating a multipart write for a named object:

```
POST
/exampleBucket/objectName?domain=yourDomainName&uploads&encoding=5:2
HTTP/1.1
 x-custom-header1-meta: value
 x-custom-headerN-meta: value
```

The custom header arguments are included because any custom metadata for the object must be declared with the initiate request. Any content in the body of the request is ignored when the multipart upload is eventually completed.

When complete, Swarm responds with a header that provides an upload ID. Record this ID so you can upload the object parts in subsequent requests.

Example of a header with an upload ID:

```
Castor-System-UploadId: VXBsb2FkIElEIG...5tMnRzIHVwbG9hZA
```

Multipart writing an immutable object

Example of initiating a multipart write for an unnamed, immutable object:

```
POST /?uploads&encoding=5:2 HTTP/1.1
 x-custom-header1-meta: value
 x-custom-headerN-meta: value
```

The custom header arguments are included because any custom metadata for the object must be declared with the initiate request. Any content in the body of the request is ignored when the multipart upload is eventually completed.

When complete, Swarm responds with a header that provides an upload ID. Record this ID so you can upload the object parts in subsequent requests.

Example of a header with an upload ID:

```
Castor-System-UploadId: VXBsb2FkIElEIG...5tMnRzIHVwbG9hZA
```

Multipart writing an existing object

In addition to creating a new object in the cluster with multipart write, you may overwrite an existing aliased or named object with multipart upload via a PUT request or add data to an existing object via an APPEND request.

The existing object remains distinct and independent until the multipart write operation is completed. When completed, the existing object is replaced by the new object. If you complete the operation with an ABORT request, the existing object remains unchanged.

Uploading the Parts

- Using object part numbers
- Uploading a part
- Uploading a part by copying from an existing object
- Validating the uploaded parts

When the initiate request is complete, you are ready to upload the individual parts of a larger object. Swarm allows any number of parts in a multipart upload. (v9.1)

To upload a part, create a POST request with the object name used in the initiate request, the upload ID returned from the initiate request, and a unique part number for each part.

If the initiate request included a domain query argument for a specific domain, the part uploads for that upload ID must also include the same domain query argument. Unlike the initiate request, the part uploads must not include an encoding query argument as they will inherit whatever was specified in the initiate request. A failed part upload can be retried without affecting the outcome of the multi-part upload.

When a part is successfully uploaded, it is stored as an immutable object whose Content-UUID is returned in the request response. Your client application must keep track of the part number used for the upload and the Content-UUID Swarm assigned it when it was stored in order to eventually complete the multipart upload, as described below.

> Note
>
> Even if the initiated object is named, each part is an immutable object that will return a Content-UUID, even though a POST on a named object does not ordinarily return that header. The parts are tenanted in the same domain as the destination object, but parts are unnamed, so they cannot reside in buckets.

Using object part numbers

Swarm uses part numbers to identify the position of each part in an object. When you upload the parts, include the upload ID and a unique part number for each part so that Swarm can assemble the parts in the correct order. You can select non-sequential part numbers for each part (for example: 2, 4, 6, 8), but Swarm will assemble the parts in sequential order.

Record each part number and its corresponding Content-UUID. This information is required to complete the multipart write procedure.

Uploading a part

To upload each part, you must include the object name or UUID returned by the initiate request, the upload ID returned from the initiate request, and a unique part number for each part that is uploaded.

```
POST /ObjectNameorUUID?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
 Host: cluster.example.com
 User-Agent: Swarm Client/0.1
 Content-Length: 43402
 Expect: 100-continue CRLF
 [ content ]
```

The content to be uploaded for the part should be in the body of the request, just like a normal POST operation.

Uploading a part by copying from an existing object

If the required parts currently exist in your storage cluster, you can create a POST request that uses the content from existing objects. When the part copy request is completed, Swarm creates an EC copy of the object for the multipart write. If the source object does not exist or cannot be read from the specified range, the request will fail. This process leaves the source and destination versions unrelated to each other.

Content Gateway

When going through Gateway, the user who is making the "PUT with copy" request must have read access to the source object.

The additional headers used in this request specify the source object and its range. The source object must be specified in the x-castor-copy-source header by UUID or object name in bucket/object-name format:

```
curl -i "$HOST/fd9cf39f056fb0dd858d8fb288c22885 PartNumber=3

&UploadID=ddd080eb400bd5531f580191e3c5a916dd66c7c1e3244dc6cad46183097677
e6dd66c7c1e3244dc6cad46183097677e60P"
 -XPOST
 -H "x-castor-copy-source: a08212d59b5bd306a52008dfef335be2"
 -H "x-castor-copy-source-range: 5-8"


HTTP/1.1 201 Created
Location: http://192.168.1.171:80/09938e338c3590b93855d7cca2179aec
Location: http://192.168.1.109:80/09938e338c3590b93855d7cca2179aec
Volume: 3f5ef63dab992ebcf28e092bb56103c3
Volume: 12e08e29145f277501a6490b602ea287
Manifest: ec
Castor-System-UploadID:
ddd080eb400bd5531f580191e3c5a916dd66c7c1e3244dc6cad46183097677e6dd66c7c1
e3244dc6cad46183097677e60P
Content-UUID: 09938e338c3590b93855d7cca2179aec
Last-Modified: Tue, 27 Sep 2016 20:49:46 GMT
Entity-MD5: 9g0GoVLSYSXc/PMI4FWKbQ==
Stored-Digest: f60d06a152d26125dcfcf308e0558a6d
Castor-System-Encoding: zfec 1.4(1, 1, 524288, 200000000)
Castor-System-Version: 1475009386.549
Etag: "4c760a34ee534bcdba91680919378e2e"
Content-Range: bytes 5-8/10
Replica-Count: 2
Date: Tue, 27 Sep 2016 20:49:46 GMT
Server: CAStor Cluster/8.2.a
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400

<html><body>New stream created</body></html>
```

If you apply a gencontentmd5 query argument or the deprecated Expect: Content-MD5 header to a part copy with a range read, the Content-MD5 is only applied to the range read.

These are the arguments and headers that are required for a part upload request that copies data from an existing object:

```
POST /ObjectNameorUUID?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
 x-castor-copy-source: uuid/name
 x-castor-copy-source-domain: domain_name
```

> Note
>
> Whenever the x-castor-copy-source header is used, the result code for the operation is in the trailing header Castor-System-Result.

-copy-source- headers

The following headers in the POST request for part uploads are optional, except for the first. If you include them, they perform the same filtering as the regular headers of those names (range, if-match, etc.), performed against the source object being copied.

> Note
>
> Error responses on conditional headers come back immediately, in place of a 202 (Accepted for processing) response. Condition failures (such as the ETags not matching) are reported in the initial HTTP response, not the castor-system-result header.

Type	Header	Notes
Source	x-castor-copy-source	Required. Must be a valid name or UUID.
Domain	x-castor-copy-source-domain	Required (unless untenanted)
Range	x-castor-copy-source-range: bytes=first-last	If the range values are out of bounds for the data, the request returns 416 (Requested Range not satisfiable).
Conditional	x-castor-copy-source-if-match: "<ETag>" x-castor-copy-source-if-none-match: "<ETag>"	The ETag must be enclosed in quotes.
	x-castor-copy-source-if-unmodified-since: <timestamp> x-castor-copy-source-if-modified-since: <timestamp>	Uses the format of the standard HTTP last-modified header.

Validating the uploaded parts

To validate the content of the uploaded part, include a gencontentmd5 query argument or Content-MD5 header in your POST argument to return a Content-MD5 header. See Content-MD5 Checksums. (v9.2)

If you intend to validate the full transfer on the complete using the Composite-Content-MD5 header, ensure that each part has an MD5 stored with it.

Completing the Multipart Write

- Requesting Completion
- Ordering the Parts
- Completion Response

You complete the multipart write procedure using a POST request with the `?uploadId=<uploadid>` query arg and no `?partNumber`. In the body of the request, include a list of all of the individual parts in JSON format with the Content-UUID and part numbers you recorded for each object when you uploaded the parts. If you include multiple parts with the same part number, the completion request will fail.

When you complete the multipart write, Swarm creates the new object or modifies the existing object by assembling the uploaded parts in ascending part-number order (the default) or by the manifest order. An existing object will be modified as specified in the initiate request, either overwriting all data with a PUT or adding the multipart upload data with an APPEND.

> Note
>
> If an object is updated during a multipart write update of the same object, the multipart write will append to the content as it exists upon completion, not as it existed when the operation was started.

Requesting Completion

The completion request must not include an encoding query argument, as it will inherit whatever was specified on the initiate request. You may provide custom metadata with the complete request and it will be merged with the metadata provided in the initiate request. If the metadata in the two requests collide, the metadata from the initiate request takes precedence.

Parallel upload complete request for 3-part object

```
POST /exampleBucket/ObjectName?uploadId=UploadId HTTP/1.1

{
   "parts": [
      {
         "partNumber": 1,
         "uuid": "12345678901234567890123456789012"
      },
      {
         "partNumber": 2,
         "uuid": "12345678901234567890123456789013"
      },
      {
         "partNumber": 3,
         "uuid": "12345678901234567890123456789014"
      }
   ]
}
```

Important

Before you complete the request, verify that the part numbers and corresponding Content-UUIDs of all of the required parts are included in the manifest. Any part not included with the manifest will not be used in the final object and will eventually be deleted by the Health Processor.

Ordering the Parts

By default, parts are ordered by part number. If you need to reassemble the parts according to the order that they appear in the manifest rather than by their part number, use the global value sortOrder set to natural (which is the manifest order):

Parallel upload complete request with manifest order

```
POST /exampleBucket/ObjectName?uploadId=UploadId HTTP/1.1

{
   "sortOrder" : "natural",
   "parts": [
         "partNumber": 2,
         "uuid": "12345678901234567890123456789013"
      },
      {
         "partNumber": 3,
         "uuid": "12345678901234567890123456789014"
      }
      {
         "partNumber": 1,
         "uuid": "12345678901234567890123456789012"
      },
      {
 ]
}
```

Completion Response

The response for a multipart write completion uses chunked transfer encoding to ensure that the client socket remains open in the event of a lengthy complete. Swarm can return an error code (400, 404, etc.) as the initial response if it finds something wrong before it starts the completion. The initial response, potentially on a eventual failure as well as a success, may be a 202 (Accepted).

- castor-system-result - In the event of a 202 (Accepted) response, the final completion status is returned in both the newline separated body data, and trailing headers, with the name `castor-system-result`.
- castor-system-error - In the event of an error, the error message is provided in the body and in a trailing header called `castor-system-error`.

Canceling a Multipart Write

You can use a DELETE request with the UploadID query argument to cancel an in-progress multipart write. This method enables the Health Processor to delete any completed or in-progress parts. Aborting a multipart write will not terminate any part uploads still in progress.

If you are canceling a multipart write that was intended to modify an existing object, the deletion only deletes the multipart write and leaves the original object intact.

Example of canceling a multipart write:

```
DELETE /ObjectName?uploadId=UploadId HTTP/1.1
```

When the cancel is completed, Swarm returns a `200 OK` code.

> Warning
> If you fail to include the `UploadID` on the abort deletion for an object that previously existed in the cluster, Swarm will attempt to delete the original object. You must include the `uploadId` query argument to ensure only the multipart write operation is deleted.

Validating a Multipart Write

Validation with Composite-Content-MD5

Swarm enables transfer validation on multipart requests by way of a composite MD5 that is computed from the Content-MD5 hashes from all of the parts. The request header that enables multipart validation is `Composite-Content-MD5`. This header provides an end-to-end integrity check of the content (excluding metadata) of a multipart write request at the time of completion. The value can be used to validate the object contents only if all of the parts were stored with a Content-MD5.

- Storage - The value of the composite MD5 is persisted and indexed in a header called Castor-System-CompositeMD5 in the metadata section of the completed object's manifest. It is not preserved across a PUT or APPEND, but it is automatically persisted across a COPY so that MD5 need not be recalculated on very large files, which is inefficient.
- Behavior - On a complete request that includes the Composite-Content-MD5 header, Swarm computes the value for the overall request from the MD5 of the concatenated Content-MD5 values stored with each part (in order).
    - If a Composite-Content-MD5 header is sent in the request, Swarm must calculate and compare it with the stored value. The complete request can succeed only if the composite value that Swarm calculates matches the value provided on the header.
    - On the completion response, the Castor-System-CompositeMD5 header is provided as a trailing header if the Content-MD5 is available on all parts, regardless of whether Composite-Content-MD5 is provided on the request.
    - For newly completed multipart writes, the Composite-Content-MD5 header is also indexed in Elasticsearch, so that it appears in listings:

```
curl 'https://www.example.com/mybucket?format=json\
 &fields=name,tmborn,etag,content-md5,Castor-System-CompositeMD
5'
[{
  "last_modified": "2017-04-08T20:37:02.868400Z",
  "castor_system_compositemd5":
"306cca04302861ed2620a328f286346f-5",
  "hash": "ae478cc4c3eb28b432825074673aeda9",
  "name": "samples/5G"
}]
```

(v9.2)

- Usage - Pass in a composite MD5 made by taking the md5 of the concatenation of the binary md5 of the parts, in order, with no gaps, providing it as the value of the 'Composite-Content-MD5' header. This triggers Swarm to collect the Content-MD5s from each part and to assemble its comparison value.
- Failure - If the calculated value does not match the supplied value, or if any part is missing a Content-MD5 header, the request fails with a 409 (Conflict). In this case, review the error message and correct the problem (such as parts missing the Content-MD5 header) before attempting the complete again.

Calculating a Composite-Content-MD5

These are different ways to represent an MD5 hash:

Base64	`rbyRpD6YijtbdFuFKakLYQ==`	Binary-to-text encoding
hexdigest	`adbc91a43e988a3b5b745b8529a90b61`	HEX string representing the hash base64.b64decode('rbyRpD6YijtbdFuFKakLYQ==').encode('hex

For Composite-Content-MD5, you need to end up with the HEX digest of the MD5 hash of the concatenated binary MD5 hashes of all of the parts, in order. The composite value starts with the hex digest of that hashed concatenation of hashes, followed by a hyphen and the number of parts:

{ hash of concatenated part MD5 hashes, in order }-{ number of parts } { hash of part1hash & part2hash & … & partnhash }-{ n }	`754e6c52092a9c1134d7f047d61db168-3`

Suppose that you have a multipart object with three parts:

- Part 1 Content-MD5 = rbyRpD6YijtbdFuFKakLYQ==
- Part 2 Content-MD5 = 9lzbDNFcX99eTYqZB4QKjg==
- Part 3 Content-MD5 = 2qHK6cuQufMzJAs6IxTmKQ==

The composite hash is this:

- `Composite-Content-MD5 = 754e6c52092a9c1134d7f047d61db168-3`

Calculating it involves this type of process:

```
partBinaryMD5-1 = base64.b64decode( contentMD5HeaderValue1 )
partBinaryMD5-2 = base64.b64decode( contentMD5HeaderValue2 )
partBinaryMD5-3 = base64.b64decode( contentMD5HeaderValue3 )
Composite-Content-MD5 = hashlib.md5("".join([ partBinaryMD5-1,
partBinaryMD5-2, partBinaryMD5-3 ])).hexdigest() + '-3'
```

Composite-Content-MD5 Example

1. Given a file divided into three parts, get the hex md5 digest for each part using `md5sum`.
    - Part 1 hash: babfc3ceb8a4568587b7d31bfff36257
    - Part 2 hash: fae6c82883c12e289bc5f12f3ecf76ef2
    - Part 3 hash: 2afdd827a9e785029f9692e82ea07cca
2. Concatenate the MD5s together into a new file.

    ```
    echo "babfc3ceb8a4568587b7d31bfff36257" >> md5sums.txt
    ```

3. See the result by catting the file:

    ```
    cat md5sums.txt
    babfc3ceb8a4568587b7d31bfff36257fae6c82883c12e289bc5f12f3ecf76ef2af
    dd827a9e785029f9692e82ea07cca
    ```

4. Convert to binary and hash it to get the composite MD5 using `xxd` and `md5sum`.

    ```
    xxd -r -p checksums.txt | md5sum
    12138b95c0af8f8e764f80d719cc7cbd -
    ```

5. Append the part count (3) to get the final composite header value: `12138b95c0af8f8e764f80d719cc7cbd-3`
6. Use the value to complete the multipart write:

```
curl -v "192.168.3.84/8c249211d4dc9683ab005760675b7c46?
 uploadId=8c249211d4dc9683ab005760675b7c460372fa38bbaad17fefc9883c48ed55
b80372fa38bbaad17fefc9883c48ed55b80P"
 -L --post301
 -H "Composite-content-MD5:12138b95c0af8f8e764f80d719cc7cbd-3"
 -d '{ "parts" : [
   { "partNumber":1, "uuid":"66d6b90b7f451f2a5ad644884d1f9106"}
   ,
   { "partNumber":2, "uuid":"5eefe725084eb6ba02cf146400f50ec4"}
   ,
   { "partNumber":3, "uuid":"64cdb1e3457a69c75ad932ad9661e93d"}
  ]}'

POST /8c249211d4dc9683ab005760675b7c46?
 uploadId=8c249211d4dc9683ab005760675b7c460372fa38bbaad17fefc9883c48ed55
b80372fa38bbaad17fefc9883c48ed55b80P
 HTTP/1.1
User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7
NSS/3.21 Basic ECC zlib/1.2.3 libidn/1.18
 libssh2/1.4.2
Host: 192.168.3.84
Accept: /
Composite-content-MD5:12138b95c0af8f8e764f80d719cc7cbd-3
Content-Length: 198
Content-Type: application/x-www-form-urlencoded

HTTP/1.1 202 Accepted
...
castor-system-result: 201
```

Validation with Content-MD5

To validate an object created by a completed multipart write, use a GET request with the `gencontentmd5` query argument and compare the result with the known value. See Content-MD5 Checksums. (v9.2)

Multipart Write Example

Following is an example of the entire multipart object uploading POST process, using CURL (with line breaks for ease of reading).

> Note
>
> **`<cluster>`** in a URL stands for **`<host>[:<port>]`**, where **`host`** is a fully qualified domain name or IP address, plus a **`port`** number if other than 80. If the Host header does not match the domain name, override it with the domain= argument.

1. Start the upload with a 0-byte POST. Set the encoding (`&encoding=2:1`) if there is no default encoding or you need a non-default setting:

   Initiate the upload

   ```
   curl -i
    "${cluster}/
    ?uploads"
    -XPOST
   ```

2. In the response, locate the headers for the UploadID and the Content-UUID, which is the new object being created:

   Returned headers

   ```
   Castor-System-UploadID:
    9282727ffcca3a09e0843281aafc13afbe5a0b53cab2e6fc4f1d24504577de4dbe
   5a0b53cab2e6fc4f1d24504577de4d0P
   Content-UUID:
    9282727ffcca3a09e0843281aafc13af
   ```

3. Write the first part using the UploadID and Content-UUID and partNumber=1:

   Upload the first part

   ```
   curl -i
    "${cluster}/c88fe8a5daf98f7ce84dc4947238f5c1
    ?partNumber=1
    &uploadid=c88fe8a5daf98f7ce84dc4947238f5c1d0b0b42b8ce464e4566cc2c0
   80ecf401d0b0b42b8ce464e4566cc2c080ecf4010P"
    -XPOST
    --data-binary
    "part 1 data"
   ```

   You will use the Content-UUID returned by this post in the manifest to complete the upload.

4. Start the second part using the UploadID and Content-UUID and partNumber=2:

Upload the second part

```
curl -i
 "${cluster}/c88fe8a5daf98f7ce84dc4947238f5c1
 ?partNumber=2
 &uploadid=c88fe8a5daf98f7ce84dc4947238f5c1d0b0b42b8ce464e4566cc2c0
80ecf401d0b0b42b8ce464e4566cc2c080ecf4010P"
 -XPOST
 --data-binary
 "part 2 data"
```

Again, you will use the Content-UUID returned by this post in the manifest to complete the upload.

5. Complete the upload with the UploadID and a part manifest:

> Note
> Be sure to follow this usage of double quotes.

Complete the upload

```
curl -i
 "${cluster}/c88fe8a5daf98f7ce84dc4947238f5c1
 ?uploadid=c88fe8a5daf98f7ce84dc4947238f5c1d0b0b42b8ce464e4566cc2c0
80ecf401d0b0b42b8ce464e4566cc2c080ecf4010P"
 -XPOST
 --data-binary '{
    "parts":[
       {
          "partNumber":1,
          "uuid":"9b149f0959839cee2c915dedfa8d7e25"
       },
       {
          "partNumber":2,
          "uuid":"76e7f0c7c417b7bca7daedbe3e18bf40"
       }
    ]
 }'
```

6. The response code on the complete will indicate success. To verify the completed object, you can HEAD the object:

Verify completed object

```
curl --head "${cluster}/c88fe8a5daf98f7ce84dc4947238f5c1"
```

Working with Policies

To make fullest use of the rich content protection features of Swarm, you will design and apply precise policies for where and how to apply protections in your implementation. Policies define how the data being uploaded to your Swarm cluster should be stored, managed, and protected.

How it works

Cluster-level policies reside in configuration settings and let you define cluster-wide and cluster-specific policies for how Swarm implements its content protection features. You can layer these baseline cluster policies with context-level (domain and bucket) policies, which reside in designated headers in the context objects themselves. Policies are not lifepoints: they do not have built-in lifetimes, nor do they specify transitions from one policy to another over time. Policies are only changed by explicit updates.

Types of content policies

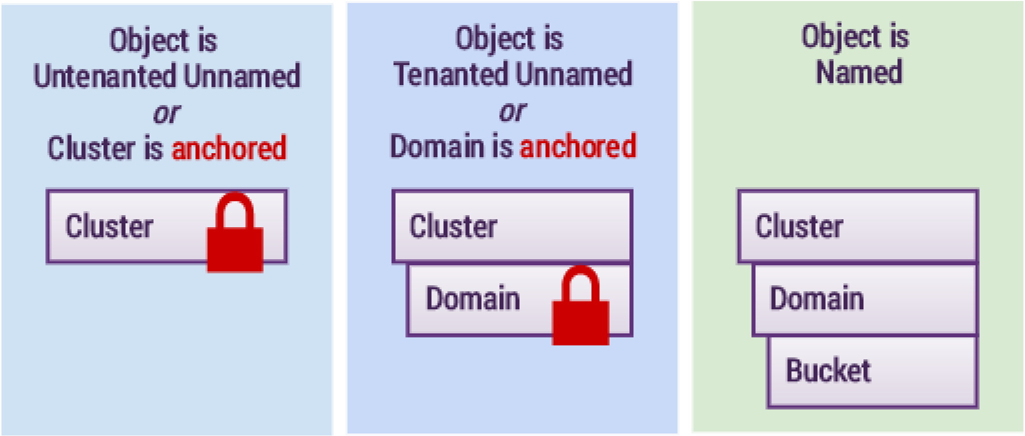
You can set policies for all of these content protection options, specific to the context you need:

- Replication - how many default, minimum, and maximum number of replicas to keep for the objects in this cluster/domain/bucket
- Erasure Coding EC - whether to enable or specify the EC encoding (`k:p`) to use for the objects in this cluster/domain/bucket
- Object Versioning - whether to enable, disable, or suspend versioning for the objects in this cluster/domain/bucket

How policies resolve

To resolve overlapping policies, Swarm begins policy evaluation by eliminating contexts (domain and bucket) that do not apply, for one of two reasons:

- because of the "`anchored`" parameter, which overrides lower-level policies
- because of the type of object, which restricts it to certain contexts:



Swarm (1) selects the anchored policy if it exists, or else (2) checks the relevant contexts from the bottom up and selects the first policy it finds.

> Note
> Within a given request to a context (domain or bucket) object, if more than one policy header of the same type is written, the last one written is honored.

Setting policies by cluster

You can set cluster-specific policies for domains and buckets by including the cluster name parenthetically and separating them with a comma. Swarm looks for and takes any cluster-specific policy values, using the default value only if it finds no match.

For example, you many want a policy (such as versioning) to be enabled in the main cluster but disabled in the target of your replication feed. This is how you enable a feature in your main cluster (myCluster) but disable it in your one-way replication cluster (myRemote):

```
Policy-{feature}: enabled (myCluster), disabled (myRemote), disabled
    or
  Policy-{feature}: enabled (myCluster), disabled
```

> Required
> Always end a multi-part policy with the default value that you prefer.

Updating policies

Swarm cluster policies are all persisted settings, so use the snmpset command on the cluster's settings object to change and persist the policies cluster-wide. Domain and bucket policies are saved as headers on those objects.

> See Using SNMP with Swarm.

Replication

- How Replication Affects Risk
- Controlling Replication Protection
- Increasing Replication Priority

- Enforcing Replicate On Write (ROW)

Swarm can provide protection on disk by creating multiple copies of each object on different nodes called replicas. You can control how many replicas are created for each object and how quickly they are created after the object is initially stored in the cluster.

> Note
>
> If one object replica exists in a cluster, there is only one instance of that object in the cluster. In this context, replica, instance, and object are all synonymous.

How Replication Affects Risk

By default, each object in Swarm is stored with two replicas, with each replica residing on a different node in the cluster. If your cluster is configured to use subclusters, replicas are distributed across subclusters.

In the event of a total failure or a hard drive fails for any reason, the cluster reacts quickly and initiates a volume recovery process for each missing drive. The recovery process rapidly creates additional replicas elsewhere in the cluster of all objects that were stored on the now missing drive(s) so that each object again has two replicas. If a second drive fails before the recovery process is completed, there is a protection risk for the only replica of the object in the cluster.

There can also be a potential period of vulnerability at the moment an object is first stored on Swarm if you do not use the Replicate On Write option to create multiple simultaneous replicas.

Controlling Replication Protection

While a rapid sequence of drive failures is unlikely, it is possible. If this presents an unacceptable risk for your application, the solution is to change your replication requirements. Changing the default replication requirements to a larger number of replicas lets you trade disk space savings for added security.

To set the replication protection for the cluster, you configure a single setting, `policy.replicas`, with three required parameters, for `min`, `max`, and `default` number of replicas:

```
policy.replicas: min=2 max=5 default=3
```

> Deprecated
>
> The cluster setting policy.replicas replaces the following three, which are all deprecated: `scsp.minReplicas` `scsp.maxReplicas` `scsp.defaultReplicas`

> See Implementing Replication Policy.

Increasing Replication Priority

By default, Swarm writes a new object to one node, responds to the application with a success code and UUID (or name), and then quickly replicates the object as needed to other nodes or subclusters. The replication step is performed as a lower priority task.

While this creates the best balance of throughput and fault tolerance in most circumstances, there are cases where you might want to give the replication task the same priority as reads and writes, which ensures replication occurs quickly even under heavy sustained loads.

Your cluster administrator can add the following parameter to the node or cluster configuration file:

```
health.replicationPriority = 1
```

With replication set to priority 1, object replication is interleaved in parallel with other operations. This might have a negative impact on cluster throughput for use cases involving sustained, heavy writes. With `health.replicationPriority = 1`, it is still possible (though much less likely) that the failure of a node or volume could cause some recently written objects to be lost if the failure occurs immediately after a write operation but before replication to another node can be completed.

Enforcing Replicate On Write (ROW)

Another replication strategy to protect your content is Replicate on Write (ROW).

Without ROW, the client writes a single copy and depends on the Health Processor (HP) to create the necessary replicas. Relying on HP leaves open a small window for data loss: the volume containing the node that holds the only copy could fail before HP completes replication. ROW eliminates that window by guaranteeing that all replicas are written on the initial request.

How it works: The ROW feature requires Swarm to create replicas in parallel before it returns a success response to the client. ROW protection applies to WRITE, UPDATE, COPY, and APPEND requests. When ROW is enabled, the secondary access node (SAN) sets up connections to the number of available peers required to create the needed replicas.

> See Configuring Replicate On Write.

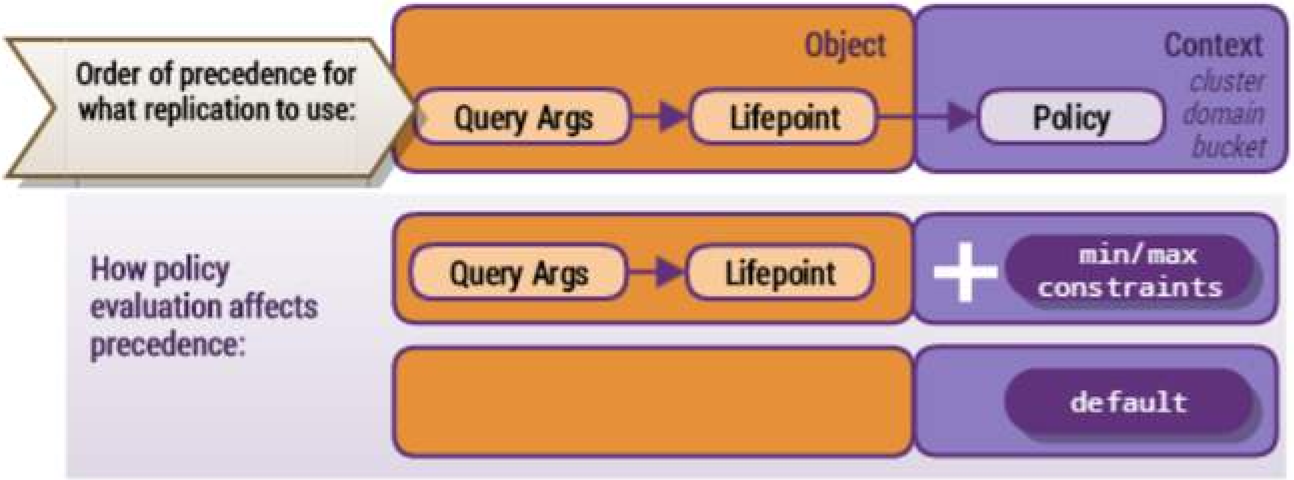
- Implementing Replication Policy

Implementing Replication Policy

- Specifying Replication
- How Replication Policies Resolve

You can create policies to control how many copies of a replicated object that Swarm will maintain, based on its context (location in the cluster). Swarm lets you maintain separate replication policies for the cluster and any of its domains and buckets.

Because Swarm lets you specify a different policy at multiple levels simultaneously, you need to understand how Swarm chooses among them in order to design your policies. As a rule, replication specified at the object level have first priority, but the context policies constrain the minimum and maximum, as well as provide the default:



> See Lifepoint Metadata Headers for setting lifepoints with replication.
> See SCSP Query Arguments for using replication query arguments.

Specifying Replication

Replication is one of several content protection features in Swarm, all of which are controlled by way of policies. Swarm evaluates policies for each object based on its cluster and context values:

- Policy-related settings for the cluster (required)
- Policy-related headers on domain and bucket objects (optional)

> For an overview of Swarm policies and how to set them at the cluster level, see Configuring Content Policies.

The replication policy has three parameters, all of which are required at the cluster level because they establish the baseline for how the cluster replicates. As with all Swarm policies, you can apply the optional anchored parameter to cluster and domains, to lock out lower-level policies.

Deprecated

The cluster setting `policy.replicas` replaces the following three, which are all deprecated: `scsp.minReplicas`, `scsp.maxReplicas`, `scsp.defaultReplicas`

Note

Replication policy does not apply to context (bucket and domain) objects themselves, which follow the cluster configuration settings `scsp.defaultContextReplicas` and `scsp.maxContextReplicas`.

You can define a different replication policy at the cluster and each context (domain and bucket) level, using this 2-part setting:

Parts	Scope	Value	Notes

replicas
required
	Affects the current context level.
	If value is missing or out of range,
	Swarm returns 400 (Bad request).
	Replaces cluster-level settings:
	- `scsp.minReplicas`
	- `scsp.maxReplicas`
	- `scsp.defaultReplicas`
		`min:`*`n`*
		`1  n  16`
			For the objects in this context, sets the lower limit for the number of replicas that must be kept.
			Overrides any policy or lifepoint header **`reps=`** constraint that is a lower value.
			Recommendations
			- To maintain just single replicas of unimportant objects, use min:1 and write those objects with reps=1, but leave the default reps at 2 or 3.
			- For reps=3, use the default `scsp.defaultROWAction = immediate`, unless the write load will be heavy and sustained, in which case protect performance by using `scsp.defaultROWAction = full`.

		`max:`*`n`*
		`min  n 20`
			For the objects in this context, sets the upper limit for the number of replicas and for k+p for each EC/ECP write request.
			Overrides any policy or lifepoint header **`reps=`** constraint that is a higher value.
			Small clusters: Set this value to the total number of nodes in the cluster, which avoids needless errors
			when lifepoint headers request more replicas than there are nodes.

		`default:`*`n`*
		`min  n max`
			For the objects in this context, sets the default number of replicas if none is specified by the current lifepoint or request.
			For most cases, **`min`** and **`default`** values match.

anchored optional	Affects contexts below the current.	none	Used with cluster and domain policies. Applies the policy to all levels below, overriding any lower-level policies.

Summary of value constraints:

```
    1  min  default  max  20
```

Example:

```
replicas min:2 default:2 max:6 anchored
```

Once you have decided the replication policy for each level, set them with the appropriate naming and method, starting with the cluster:

Scope	Type	Name, SNMP MIB	Configuration Method
cluster	Configuration setting	`policy.replicas` `policyReplicas`	Cluster policies are captured in a single setting that includes all three required parameters, for a baseline: `policy.replicas: min:2 max:16 default:2` Use the snmpset command on the cluster's settings object to change and persist the cluster-wide policies. See Using SNMP with Swarm.
domain and bucket	Persisted header	`Policy-Replicas` `policyReplicas`	Context (domain and bucket) policies are captured in persisted headers that can override some or all of the cluster policies. Use a write method (POST for new, COPY for update) on the domain object to set its policy values. Use a read method (GET, HEAD) on the domain object to check its policy values.

Replication is subject to policy resolution on each write request. For contexts, Swarm uses the policy at the given level unless a level above it is anchored. In that case, Swarm uses the policy that is anchored.

In this example, this new domain has its own required policy for named objects (which bucket owners will not be able to override). Tenanted unnamed objects follow this policy regardless of "anchored":

Writing a domain policy

```
curl -iL -XPOST --post301 --data-binary ""
 -H "Policy-Replicas: max:6 default:4 anchored"
 "http://{cluster}?domain=myDomain"
```

In this example, this new bucket increases the maximum, but note that this policy will be canceled by any policies "`anchored`" above it at the domain or cluster level:

Writing a bucket policy

```
curl -iL -XPOST --post301 --data-binary ""
 -H "Policy-Replicas: max:9"
 "http://{cluster}/myBucket?domain=myDomain"
```

How Replication Policies Resolve

If a content object is subject to a set of conflicting policies, the goal of policy resolution is to determine the correct number of replicas to keep for that object. This evaluation must be bounded by a minimum and maximum, and, as with all types of Swarm policies, must honor the anchored option, which inhibits evaluation at lower levels.

Swarm evaluates replication policy according to scope, which is tied to object type (because types can only be stored in a certain type of context):

Scope	Object type	Min = highest among:	Max = lowest among:	Default = first found, constrained by min/max:
Cluster	• Untenanted unnamed	• Cluster setting	• Cluster setting	1. Lifepoint 2. Cluster setting
Domain	• Tenanted unnamed	• Cluster setting • Domain object header	• Cluster setting • Domain object header	1. Lifepoint 2. Domain header 3. Cluster setting
Bucket	• Named	• Cluster setting • Domain object header • Bucket object header	• Cluster setting • Domain object header • Bucket object header	1. Lifepoint 2. Bucket header 3. Domain header 4. Cluster setting

Every cluster update must supply all three values (min, max, default), but any subset is allowed for domain, bucket, and unnamed. They can be in any order, and any case.

Affected Objects		Example	Effect
cluster	Untenanted unnamed	`policy.replicas: min:2 max:10 default:4`	Constrains the lifepoint value for `reps`: 2 ≤ reps ≤ 10. If `reps` is unspecified, defaults to 4.
domain	Tenanted unnamed	`Policy-Replicas: min:1 max:8`	Constrains the lifepoint value for `reps`: 1 ≤ reps ≤ 8. If `reps` is unspecified, defaults to 4 (from the cluster setting).
bucket	Named	`Policy-Replicas: default:2 max:6`	Constrains the lifepoint value for `reps`: 1 (per domain setting) ≤ reps ≤ 6. If `reps` is unspecified, defaults to 2.

Conflict Example

Policies at different levels can conflict, such as in this example:

First evaluation	= Invalid

Cluster: min:1 max:10 ● Domain: min:2 max:5 ● Bucket: min:6 max:8	Most constrained values: ● min:6 max:5

The most restrictive min is 6, and the most restrictive max is 5, which is not a valid pair, because the max is lower than the min. Given such a conflict, Swarm logs a warning message and reevaluates the policy, omitting the lowest level in the hierarchy (bucket, in this example), until the conflict is resolved. The lowest level is dropped because Swarm privileges the values of the cluster owner over those of the domain owner, over those of the bucket owner.

First evaluation	= Invalid	Second evaluation	= Valid
Cluster: min:1 max:10 ● Domain: min:2 max:5 ● Bucket: min:6 max:8	Most constrained values: ● min:6 max:5	Cluster: min:1 max:10 ● Domain: min:2 max:5 ● ~~Bucket: min:6 max:8~~	Most constrained values: ● min:2 max:5

> Note
> Default values specified at lower levels override the default values set at higher levels, as long as they fall within the resolved min/max values.

Erasure Coding EC

Replication is a proven and valuable mechanism to ensure data integrity, but the cost per GB of storage can get high as object sizes and cluster sizes grow. A complementary data protection strategy, erasure coding (EC), provides high data durability with a smaller footprint. Swarm manages EC and replication together to optimize cost-effectiveness, converting objects between them seamlessly and dynamically, based on the policies that you set.

- How EC works
- How much EC protects
- How much drive space EC saves

How EC works

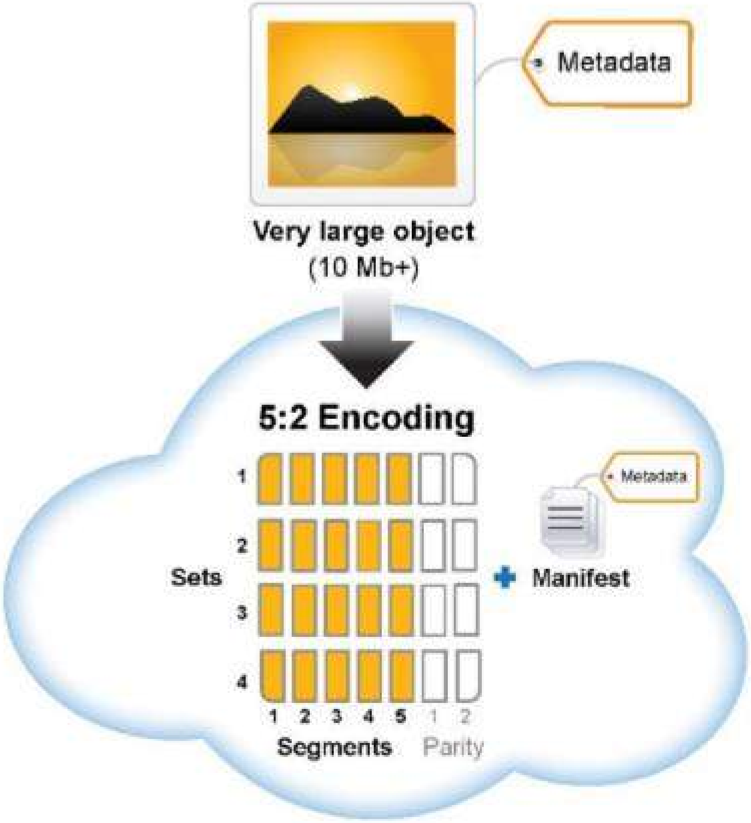
Erasure coding breaks the original object into multiple data segments (k) and computes additional parity segments (p) based on the content of the data segments. This results in m total segments (k + p = m) being distributed to m different nodes (or subclusters) in the storage cluster (see the ec.protectionLevel setting).

The erasure coding encoding level is expressed as a tuple in this format:

```
{data segments}:{parity segments}
```

For very large objects, Swarm creates multiple sets of erasure segments. The object breakdown into one or more erasure sets is transparent to external applications. A GET or HEAD of an erasure-coded object uses the same syntax as a replicated object.

The following illustration represents how erasure coding works:



How much EC protects

If a hard drive or a node containing an erasure segment fails, Swarm can still read the object as long as there are still k total segments (any combination of original data or parity) remaining in the cluster. In other words, the protection against drive failure for the object is equal to the number of specified parity p segments.

For example, because the segments from a 5:2 (5 data segments with 2 parity segments for a total of 7 segments) or 8 :2 (8 data and 2 parity segments for 10 total segments) erasure code are distributed to different nodes, they are protected against the loss of any two nodes. An erasure-coded object is immediately retrievable when accessed even if some segments are missing. However, regenerating the missing erasure set segments is still performed in a self-healing, cluster-initiated manner (similar to the recovery process for replicated objects) to protect against further drive loss. This process kicks off automatically when a missing volume is detected and automatically regenerates any missing segments.

How much drive space EC saves

The amount of drive space (or footprint) used for erasure-coded objects depends on the ratio of data to parity segments in the specified encoding.

Use the following formula to roughly calculate the drive space that you can expect to see used by an EC object with one set of erasure segments:

(total segments ÷ data segments) × object size	= object footprint

((k+p) ÷ k) × GB	= total GB

How footprint changes with different EC encoding (versus 3 reps)

- 1 GB object with 5:2 encoding: ((5 + 2) ÷ 5) × 1 GB = 1.4 GB (vs. 3 GB for replication)
- 3 GB object with 5:2 encoding: ((5 + 2) ÷ 5) × 3 GB = 4.2 GB (vs. 9 GB for replication)
- 3 GB object with 7:3 encoding: ((7 + 3) ÷ 7) × 3 GB = 4.3 GB (vs. 9 GB for replication)

Note
Additional system metadata is written with each EC segment, which adds about 16 bytes per segment.

> See Hardware Requirements for Storage for how to size and optimize your cluster hardware for erasure coding.

- Implementing EC Encoding Policy
- Methods Affected by Erasure Coding
- Conversion between Content Protection Types
- Troubleshooting Erasure Coding

Implementing EC Encoding Policy

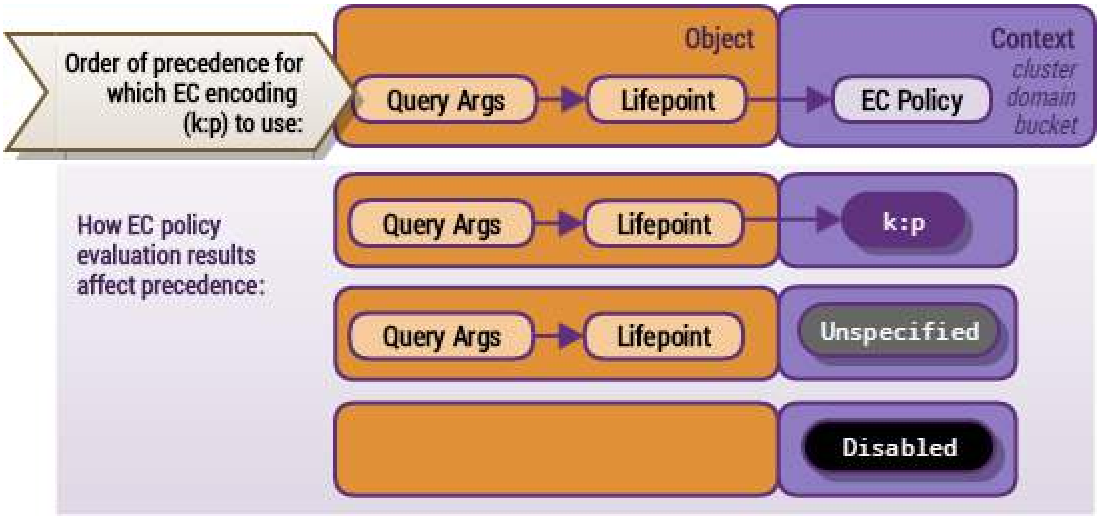
- Options for specifying EC encoding
- How Swarm resolves EC policies
- Example EC policy resolution
- How to view and set EC encoding policy

You can create context-level policies to manage EC (erasure coding) across the cluster. Swarm ships with cluster-level EC encoding unspecified, which means that every object is fully replicated, regardless of its size, unless EC is specified at the object level.

As of Swarm 8.1, you can create custom EC policies for encoding (`k:p`) and object size minimums that you can apply to specific domains and buckets (contexts). Context-level policies let you build a wide range of erasure-coding controls to meet your business needs:

- Disable EC on a bucket, forcing it to perform full replication on every object
- Quadruple the object size that will trigger erasure coding in a special-use bucket
- Set different EC encodings for named versus unnamed objects in a specific domain
- Lock the domain policy as "unspecified" to cancel any policies that bucket owners may have set, forcing them to accept the current cluster policy
- Lock the cluster policy as "disabled" to stop all erasure coding, voiding any query argument or lifepoint encodings passed in with the request

Because Swarm lets you specify a different encoding (`k:p`) at multiple levels simultaneously, you need to understand how Swarm chooses among them in order to design your EC policies. As a rule, encoding specified at the object level wins over any held by its context; however, the context policy evaluation can disable the encoding requested at the object level:



An individual object can specify EC encoding in its lifepoint header, as `reps=k:p`. To override that lifepoint on an SCSP POST or PUT method, you can add a query argument, as `encoding=k:p`. Adding the optional `erasurecoded` argument overrides the active `policy.ecMinStreamSize` requirement, but it cannot override an EC policy that evaluates to "`Disabled`".

> See Lifepoint Metadata Headers for setting lifepoints with erasure coding.
> See SCSP Query Arguments for using erasure coding query arguments.

Options for specifying EC encoding

You can define a different EC encoding policy at the cluster and each context (domain and bucket) level, using this 2-part setting:

Part	Scope	Value (string)	Notes	Example
Encoding	Affects the current context level. If value is missing or out of range, returns 400 Bad request	unspecified (default)	Allows erasure coding at this level and below, using the available policy or cluster defaults.	`unspecified`
		disabled	Disables erasure coding at current level only.	`disabled`
		k:p {data segments}:{parity segments}	The specific EC encoding to use at this level. Requirements: • 1 ≤ k • ec.minParity ≤ p • k + p <= policy.replicas max Subject to the resolved policy's eCMinStreamSize value unless it is overridden by the erasurecoded arg.	`7:3`

Lock (optional)	Affects any contexts below the current context.	anchored	Applies this encoding to all levels below, overriding any lower-level policies.	`disabled anchored`
				`7:3 anchored`

EC encoding is subject to policy resolution on each write request, evaluating the object-level encoding with that of its context. For contexts, Swarm uses the policy at the given level unless a level above it is anchored. In that case, Swarm uses the encoding that is anchored.

> Tip
> The "`anchored`" lock exists to let you suppress any policies that may have been added by domain or bucket owners below.

Anchoring lets you enforce these two situations:

- Disable erasure-coding cluster-wide:

```
policy.ecEncoding = disabled anchored
```

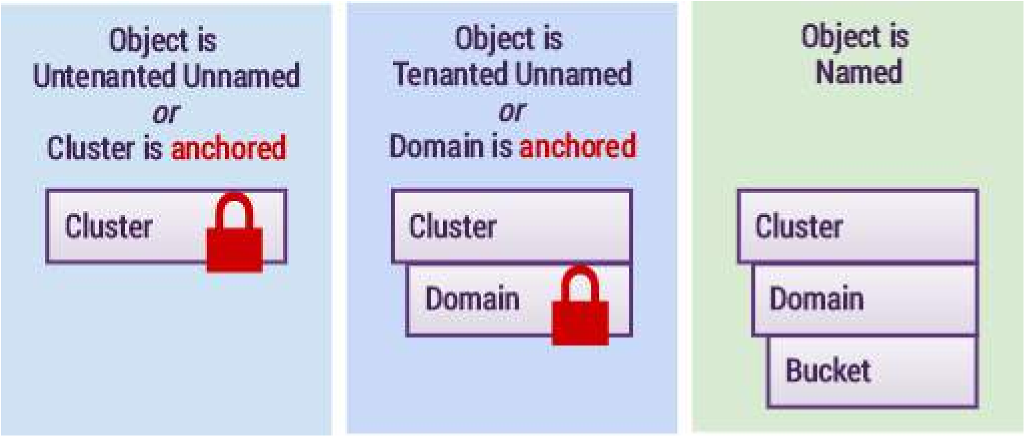
- Require objects to specify encoding, regardless of context:

```
policy.ecEncoding = unspecified anchored
```

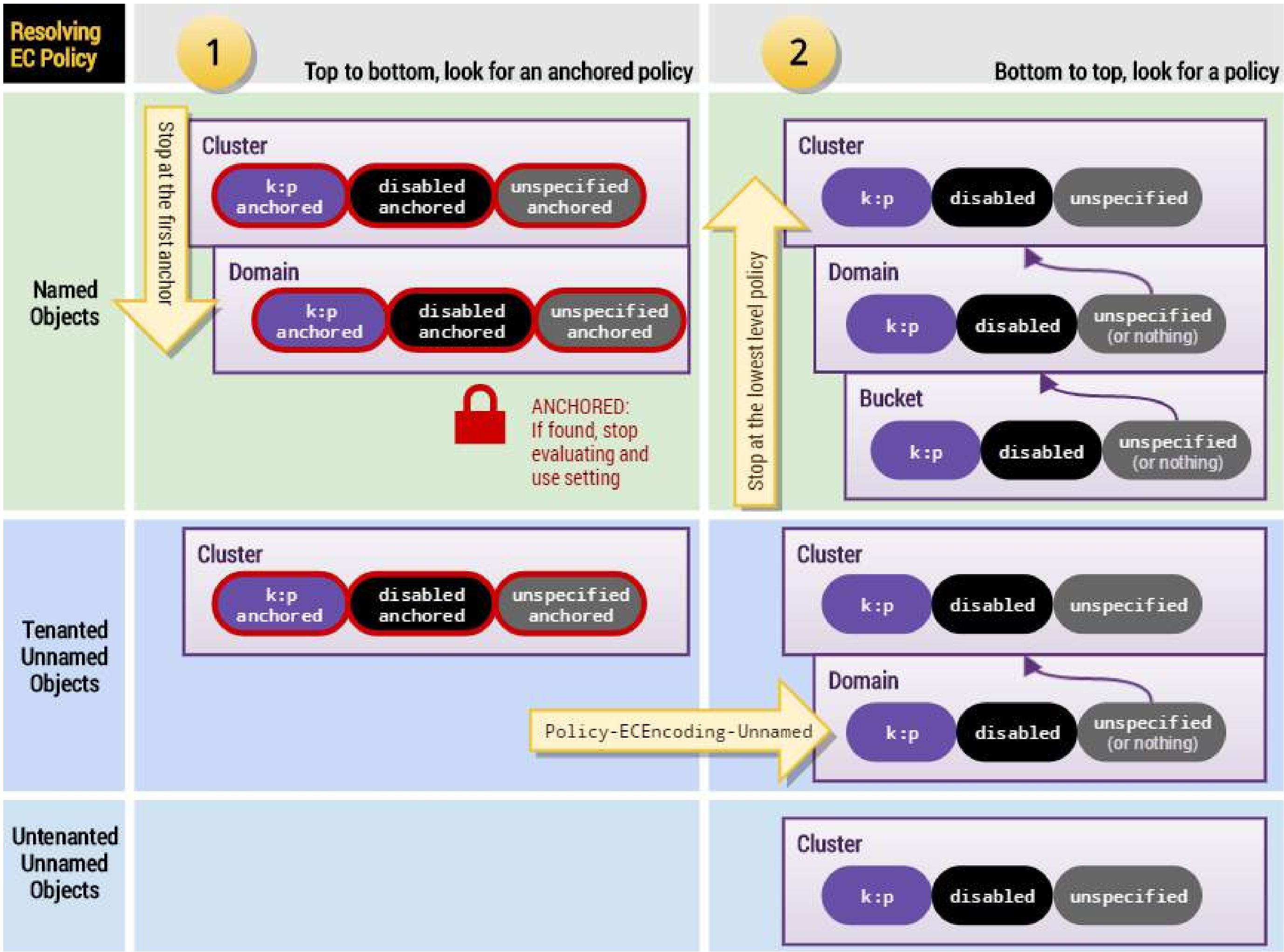
How Swarm resolves EC policies

Swarm begins EC policy evaluation by eliminating contexts (domain and bucket) that do not apply, for one of two reasons:

- because of the type of object
- because of the "`anchored`" override


Object is Untenanted Unnamed or Cluster is anchored
Cluster
Object is Tenanted Unnamed or Domain is anchored
Cluster
Domain
Object is Named
Cluster
Domain
Bucket

Swarm (1) selects the anchored policy if it exists, or else (2) checks the relevant contexts from the bottom up and selects the first policy it finds:



Example EC policy resolution

For example, consider a set of conflicting context-level policies such as this:

cluster		Configuration setting	`policy.ecEncoding`	`7:3`

	domain	Persisted header	`Policy-ECEncoding`
			`Policy-ECEncoding-Unnamed`
				`5:2 anchored`
				`5:2`

	bucket	Persisted header	`Policy-ECEncoding`	`disabled`

The EC policy evaluation resolves differently depending on the type of object being written:

Object Type	Evaluates to	Explanation

named	`5:2`	The bucket setting is ignored because the domain setting is anchored.
tenanted unnamed	`5:2`	The bucket setting is ignored because unnamed objects cannot reside in a bucket. If the `Policy-ECEncoding-Unnamed` domain header was unspecified, it would have defaulted to the cluster value.
untenanted unnamed	`7:3`	The domain setting is ignored because untenanted objects cannot reside in a domain.

Tip
To see how Swarm resolves the protection policy (whether to use erasure coding or replication) on a request, see the diagram in Troubleshooting Erasure Coding.

How to view and set EC encoding policy

EC Encoding is one of several content protection features in Swarm, all of which are controlled by way of policies. Swarm evaluates policies for each object based on its cluster and context values:

- Policy-related settings for the cluster (required)
- Policy-related headers on domain and bucket objects (optional)

> For an overview of Swarm policies and how to set them at the cluster level, see Configuring Content Policies.

Creating a policy involves changing one or both EC requirements for the given scope:

1. ecEncoding - Whether to allow/prevent/change the EC encoding
2. ecMinStreamSize* - Whether to change the object size that triggers erasure-coding.

Important
For efficiency, ecMinStreamSize* is given in units of nMB and nGB, not bytes, as were the old settings. Be sure to convert (not copy) your old ec.minStreamSize value to the new setting that specifies units. Note that 1 MB = 1,000,000 bytes.

Exception
If the request is to initiate a parallel upload (which must be erasure-coded), then
the ecMinStreamSize limit does not apply.

Once you have decided the encoding policy for each level, set them with the appropriate naming and method, starting with the cluster:

Scope	Type	Persisted Name, SNMP MIB	Configuration Method

cluster	Configuration setting	policy.ecEncoding, policyECEncoding policy.ecMinStreamSize, policyECMinStreamSize	Set the cluster parameters that are specific to EC encoding policy and verify the other EC-related settings. Use the snmpset command on the cluster's settings object to change and persist the cluster-wide policies. See Using SNMP with Swarm. See Configuring Content Policies for details.
domain	Persisted header	Policy-ECEncoding, policyECEncoding Policy-ECMinStreamSize, policyECMinStreamSize Policy-ECEncoding-Unnamed, policyECEncodingUnnamed Policy-ECMinStreamSize-Unnamed, policyECMinStreamSizeUnnamed	Use a write method (POST for new, COPY for update) on the domain object to set its policy values. Use a read method (GET, HEAD) on the domain object to check its policy values. Because unnamed objects are tenanted directly in domains, you can specify separate encoding for those objects, as if they were in their own bucket: ● Set `Policy-ECEncoding-Unnamed` to allow/prevent/change erasure-coding of unnamed objects in this domain. ● Set `Policy-ECMinStreamSize-Unnamed` to change the size that will trigger erasure-coding of unnamed objects in this domain.
bucket	Persisted header	Policy-ECEncoding, policyECEncoding Policy-ECMinStreamSize, policyECMinStreamSize	Use a write method (POST for new, COPY for update) on the bucket object to set its policy values. Use a read method (GET, HEAD) on the bucket object to check its policy values.

In this example, this new domain has its own required policy for named objects (which bucket owners will not be able to override), but unnamed objects will follow the current cluster-level policy:

Writing a domain policy

```
curl -iL -XPOST --post301 --data-binary ""
 -H "Policy-ECMinStreamSize: 2MB"
 -H "Policy-ECEncoding: 7:3 anchored"
    -H "Policy-ECEncoding-Unnamed: unspecified"
 "http://{cluster}?domain=myDomain"
```

In this example, this new bucket disables erasure-coding, but note that this policy will be canceled by any policies "`anchored`" above it at the domain or cluster level:

Writing a bucket policy

```
curl -iL -XPOST --post301 --data-binary ""
 -H "Policy-ECEncoding: disabled"
 "http://{cluster}/myBucket?domain=myDomain"
```

Methods Affected by Erasure Coding

Following is how the SCSP methods are affected by erasure-coded objects:

Method	Description
POST	A new object written to the storage cluster is erasure-coded if it meets the EC criteria.
PUT	As with POST, if a non-erasure-coded object is PUT and the data causes the object to meet the EC criteria , the object can be erasure-coded.
APPEND	As with PUT, if data appended to a non-EC object causes the object to meet the EC criteria, the object can be erasure-coded. APPENDs are optimized to POST a new set of EC segments with the appended data and PUT the manifest for a previously erasure-coded object, instead of rewriting the whole object to include the appended data as with replicated objects. Data appended to an object that was not previously erasure-coded can cause the object to become erasure-coded if the additional appended data pushes the object size over the configured `ECMinStreamSize` threshold. Attempting to use the argument `erasurecoded` without `encoding` or when the cluster is not configured for EC will result in a 400 Bad Request error. On an APPEND for an existing EC object, the new segments will be encoded with the parameters of the first existing erasure set.
COPY	Instead of rewriting the entire object to PUT the metadata, COPY for erasure-coded objects PUTs the metadata on the manifest only without rewriting any content data.
HEAD	Returns the Manifest: ec header for an erasure-coded object.
GET	The cluster automatically retrieves all segments, so a GET does not behave differently for an erasure-coded object.
DELETE	Swarm deletes both the manifests and the segments for the object.

Conversion between Content Protection Types

- Lifepoint conversions
- Encoding conversions

- Using cache coherency headers after erasure coding

Lifepoint conversions

The Health Processor maintenance mechanism converts objects as necessary between:

- Standard replication and erasure coding
- Erasure coding and standard replication

When you configure your cluster settings for this type of conversion, the Health Processor replicates the logical object back into the cluster with the new replication scheme. After the conversion, the object is visible with some header changes, and the new object version supersedes the older version. If the conversion is performed in a source cluster, it is re-replicated to the target cluster as a new version of the object.

Content protection can change due to a set of explicitly specified lifepoint policies over time. An explicit lifepoint specification that includes a reps= value—either as a whole number or a colon-separated k:p EC encoding—takes precedence over any default cluster setting. This explicit conversion responds to explicit lifepoint specifications. Explicit conversions include replication to erasure coding, erasure coding to replication, and erasure coding to another erasure-encoding scheme.

Explicit conversions occur on the next Health Processor cycle following the lifepoint change.

Encoding conversions

In addition to lifepoint conversions, the Health Processor performs encoding conversions when:

- The cluster includes a policy.ecEncoding value, which sets the number of data and parity segments to be used when erasure coding objects (for example. 5:2).
- The object is larger than the policy.ecMinStreamSize value, which indicates the minimum size (in bytes) for an object to be automatically erasure-coded.

If the object is replicated wholly, policy.ecEncoding is specified, and the object size is greater than the policy.ecMinStreamSize value, the object will be converted to erasure coding. This implicit conversion occurs because of your cluster settings. Implicit conversions are used to convert legacy data—perhaps without lifepoints—to the default cluster encoding scheme, enabling legacy data to take advantage of the new capability. However, if the object is replicated and policy.ecEncoding is not configured or the object size is less than the policy.ecMinStreamSize value, the object remains replicated at scsp.defreps replicas.

The Health Processor converts these objects at a slower rate than the next Health Processor cycle to balance its processing cycles between object conversions and other system requirements. The ec.conversionPercentage setting governs the conversion rate. The ec.conversionPercentage setting defaults to zero, which implies no implicit conversion. Until the object is converted, it will be replicated with scsp.defreps reps.

Using cache coherency headers after erasure coding

When the Health Processor converts objects from standard replication to erasure coding, it replaces the Etag and the Castor-System-Version header. When the if-match and if-none-match cache coherency headers compare the request header content against the Castor-System-Created header on the object (which does not change during the conversion), the headers will believe the object has changed, even though the content data is actually the same. As a result, the cache coherency headers will not always behave as expected.

To achieve consistent results,

- Use if-modified-since and if-unmodified-since headers after erasure-coding conversion or during remote replication where either the original or destination object is erasure-coded.

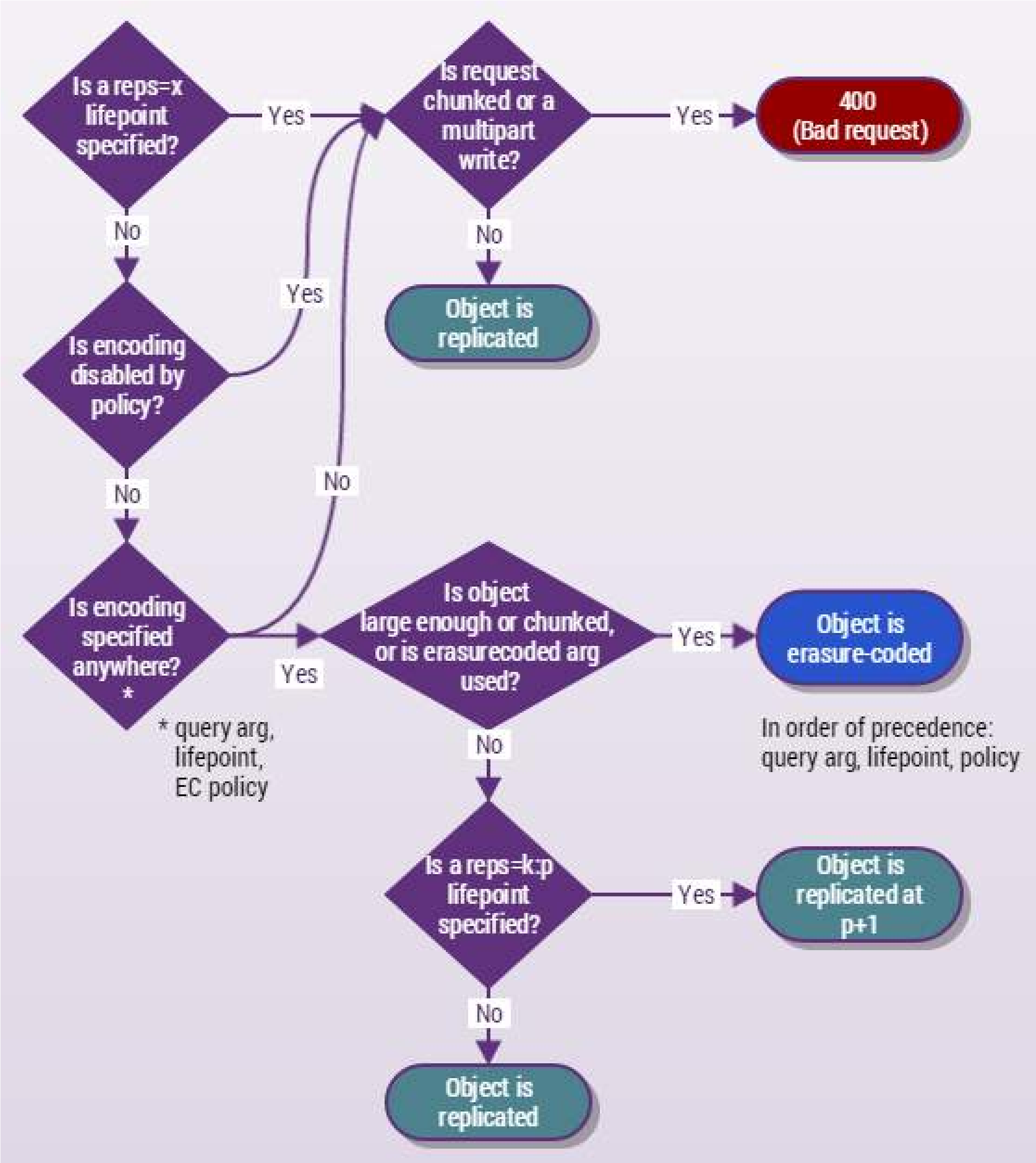
- Do not use only the Etag header after an erasure-coding conversion because the header may inadvertently change during an object conversion.

Troubleshooting Erasure Coding

- Bad Request
- Replicated, Not Erasure-Coded

Tip
When troubleshooting encoding and policy issues, add the `verbose` query argument, which will return all of the relevant headers in the response.

The following diagram details how Swarm evaluates protection requests:


Is a reps=x lifepoint specified?
Yes
Is request chunked or a multipart write?
Yes
400 (Bad request)
No
No
Yes
Object is replicated
Is encoding disabled by policy?
No
No
Is encoding specified anywhere? *
Yes
Is object large enough or chunked, or is erasurecoded arg used?
Yes
Object is erasure-coded
* query arg, lifepoint, EC policy
No
In order of precedence: query arg, lifepoint, policy
Is a reps=k:p lifepoint specified?
Yes
Object is replicated at p+1
No
Object is replicated

> Note
>
> Every multipart write must be erasure coded for upload; however, if the uploaded object does not meet the current policy for EC encoding, the HP will convert it to a replicated object. To maintain erasure coding for the lifetime of the object, add a lifepoint to that effect.

Bad Request

Both chunked requests and Multipart Write operations must be erasure-coded to succeed. A 400 (Bad Request) results from such a request having one of these problems:

- The request includes a `reps=x` lifepoint (which specifies number of replicas, not encoding).
- Encoding is disabled, which overrides any query arg or lifepoint encoding.
- Encoding is not specified anywhere.

Replicated, Not Erasure-Coded

If encoding is allowed, and specified in the lifepoint, but the object is too small to qualify, it will be replicated at **`p+1`**. For example, if 5:2 encoding was specified, 3 replicas will be written.

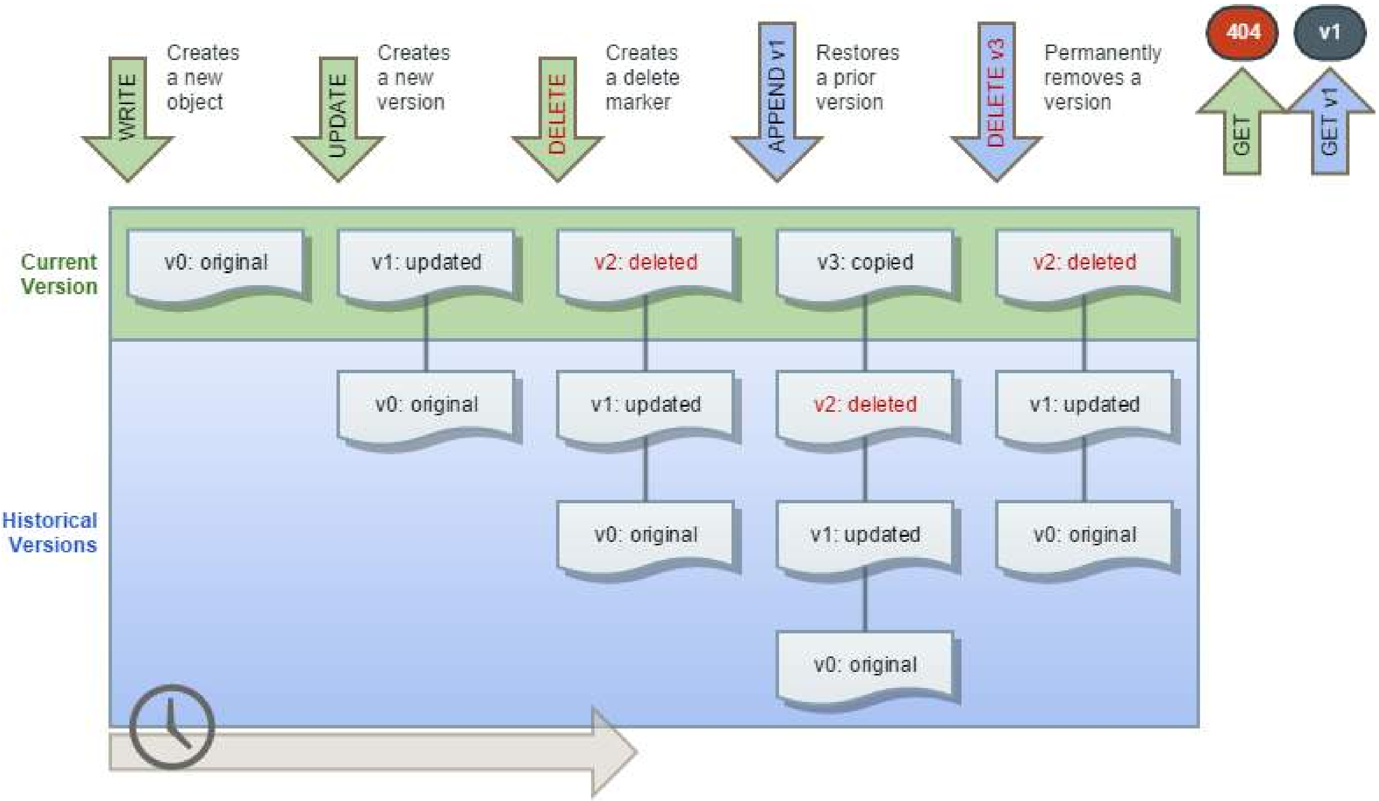
Object Versioning

- S3 Versioning
- Why use versioning?
- What gets versioned

Object-level versioning is a powerful content protection option that tracks, secures, and provides access to historical versions of objects, even after they are deleted. With versioning, your applications can read, list, revert, and purge prior versions as well as restore objects that were deleted by mistake.

> Note
> Using Swarm versioning with SCSP operations has no dependencies. To use Swarm versioning with Amazon S3, you must run Content Gateway version 4.1 or higher.

Versioning preserves a set of historical variants of an object, the original plus subsequent updates to it, up to and including the latest version:



These are key capabilities of Swarm versioning:

- Unlimited versions – The number of supported versions for a given object is unbounded, and all versions have a unique version ID. You can list all versions, and you can access, restore, and permanently delete specific versions via the version ID.
- Flexible policy – The cluster administrator changes the cluster policy settings to allow versioning; the domain administrator can then allow and even require versioning in that domain. If allowed by the cluster and domain, a bucket owner can enable/disable versioning for a specific bucket.
- Lossless concurrent updates – Swarm captures simultaneous PUT updates and resolves the order in the version chain. Swarm preserves all versions, even those overlapping in time, with the latest update as the current version.
- Accurate disk reporting – Each object revision in a domain/bucket with versioning-enabled preserves and reports its full size on disk. Swarm includes all object revisions in its 'du' responses, if requested, which means the size for deleted and historical versions counts towards bucket and domain totals.
- Support for search and replication – Swarm Versioning works with both Search feedsand Replication feeds, provided that all clusters are running the same version of Swarm.

Content Router

Content Router is incompatible with Swarm versioning and has been removed from version 8.0.

S3 Versioning

Swarm's native object versioning feature is interoperable with AWS S3 versioning. The implementation includes these improvements:

- Ability to disable versioning:
  AWS S3 only allows for versioning to be suspended once enabled on a bucket. Swarm provides the ability to disable versioning and automatically clean up the prior versions in order to reclaim storage space.
- Delete marker consolidation:
  Unlike AWS S3 where continued DELETE operations on a deleted object will record additional delete markers in the version history, Swarm will acknowledge the subsequent deletes without recording additional delete markers. Multi-factor authentication delete is not supported.
- Expanded version listing:
  Swarm supports version listing batches up to 2000 items while AWS S3 limits these listing results to batches of 1000. Additionally, Swarm does not break batches on version boundaries. Delimiter case is currently not supported for version listing.
- Simplified ACL management:
  When using per-object ACLs with versioning, the ACL for the current version of the object applies for determining authorization. To change the ACL for an object's entire version chain, update the object without spe cifying a version.

Why use versioning?

Versioning meets two key needs:

- You need extremely durable data retention and archiving.
- You need to be able to recover when data is erroneously overwritten or deleted.

With versioning enabled, you can retrieve and restore any of the prior versions of a stored object, which lets you recover from data loss, whether caused by user error or application failure:

- Deleting an object – Instead of removing it permanently, Swarm inserts a delete marker, which becomes the current object version. You can still restore any previous version.
- Overwriting an object – Swarm performs the update by creating a new version, which means that you can roll back a bad update by restoring the previous version.

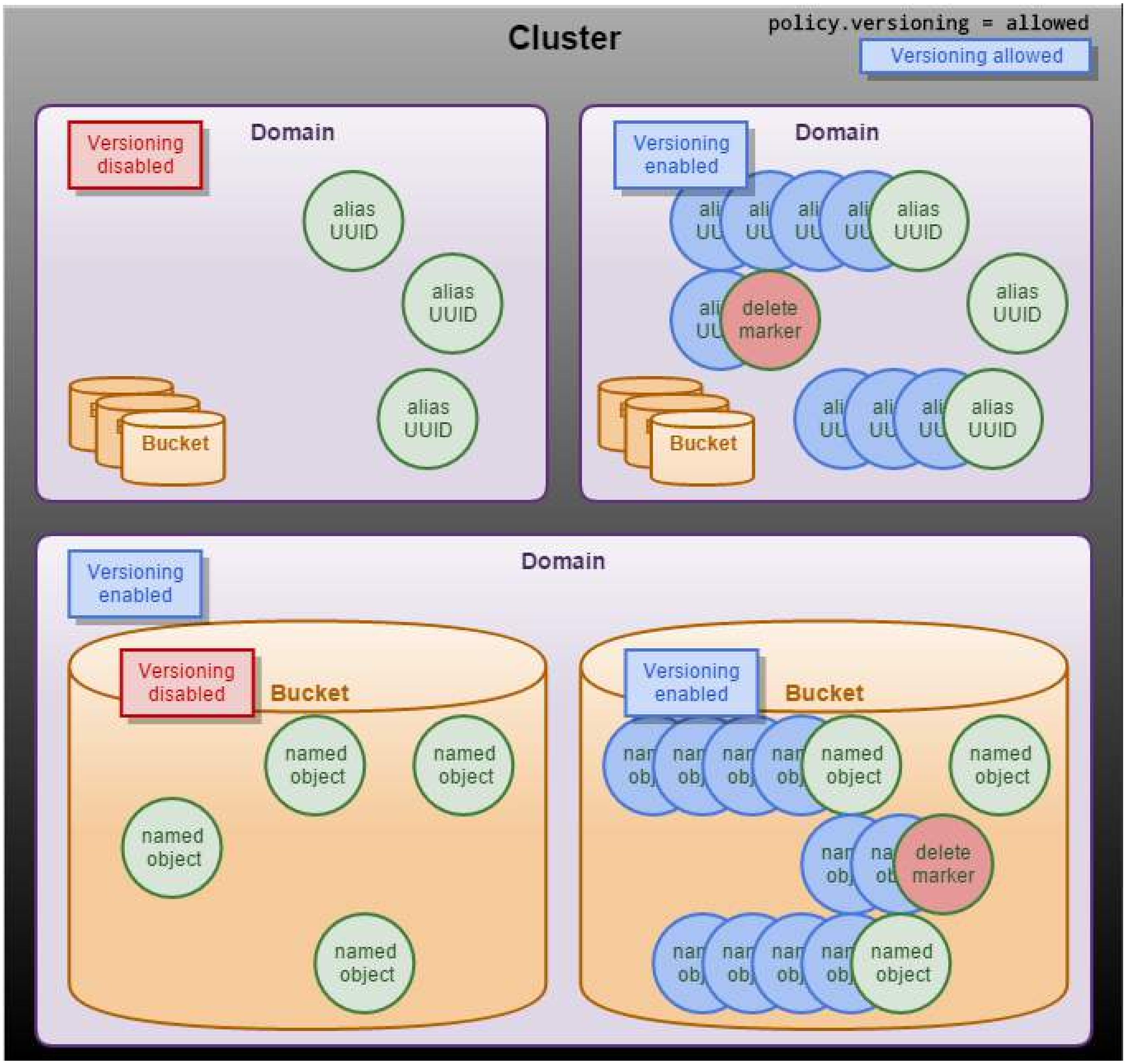
By default, versioning is disabled across the cluster. To avoid excessive storage usage, enable versioning in a targeted way, only where change control is required.

What gets versioned

Choosing to use versioning means that you get the ability to preserve, retrieve, and restore every update of every object that you store in that context (domain or bucket). With Versioning, Swarm archives another copy of an existing object whenever you update or delete it. GET requests retrieve the most recently written version, but you can get back the older versions of an object by specifying a version in the request.

Once you configure the cluster to allow versioning, administrators can selectively enable versioning at

- the domain level (for alias objects)
- the bucket level (for named objects)



When you DELETE a versioned object, Swarm creates a delete marker so that subsequent simple (unversioned) requests will no longer retrieve the object. However, Swarm still stores all versions of that object, so you can retrieve and restore it, if need be.

Note that these types of Swarm objects cannot be versioned:

- Domains
- Buckets
- Unnamed objects (which are immutable)
- Alias objects not tenanted in a domain

Only objects are versioned, not domains or buckets (contexts). This means that, if you accidentally delete a bucket, the bucket is lost; however, Swarm pauses the recursive delete of the bucket's contents for the duration of your grace period (`health.recursiveDeleteDelay`). You have time to recreate the bucket with the same headers and so not lose any data (see Restoring Domains and Buckets). If you choose not to restore the bucket and the grace period expires, the content will start to disappear as Swarm's HP begins cleaning up all versions of the obsolete content, to reclaim space.

- Implementing Versioning
- Working with Versioning
- Versioning Operations
- Versioning Examples

Implementing Versioning

Swarm Versioning is an optional feature introduced in release 8.0; it is modeled on Amazon S3 Versioning to extend Gateway's support for S3.

- Guidelines for Versioning
- Configuring Versioning
- Managing Polices across Clusters

Guidelines for Versioning

Review these guidelines before implementing versioning in your cluster:

- Plan for higher disk utilization with versioning: each update to a versioned object adds a new object to the cluster (one object updated twice results in three objects stored).
- Make use of lifepoints to control the lifetime — and thus the cost of storing — multiple versions of your objects.
- For resource management, limit versioning to the specific domains and/or buckets for which it's needed.
- For best performance, be sure to enable the Overlay Index on your cluster.
- When replicating between clusters that both enable versioning, avoid unnecessary updates to versioning policies: new policies cannot take effect in the remote cluster until the domain or bucket is replicated, which might take a while.
- Legal hold buckets cannot be versioning-enabled. If you are using legal hold, note that you need to use the ETag header value (which is the version identifier) to hold a previous object version.
- Upgrades: Versioning is integrated with Swarm and available on upgrade, but take these steps before enabling versioning:
    - Complete the upgrade of both the storage cluster and the search cluster .
    - If you are using replication feeds, upgrade both the source and target clusters to the same release of Swarm before enabling versioning.

Configuring Versioning

Versioning is one of several content protection features in Swarm, all of which are controlled by way of policies. Swarm evaluates policies for each object based on its cluster and context values:

- Policy-related settings for the cluster (required)
- Policy-related headers on domain and bucket objects (optional)

> For an overview of Swarm policies and how to set them at the cluster level, see Configuring Content Policies.

Context

The versioning state of a given object depends on its context:

- Alias object — context = cluster + domain in which it's tenanted
- Named object — context = cluster + domain + bucket

> Important
> Untenanted alias objects cannot be versioned. Alias objects are always tenanted if `cluster.enforceTenancy` is enabled in the cluster (which is required for Gateway).

Policy

Swarm uses the versioning policy (set via cluster settings and domain/bucket headers) to determine what versioning-related operations are allowed on each object. The versioning state of the immediate context applies to ever y object in that context, without exception.

Each alias or named object has one of three versioning states:

disabled	(default) No versioning exists, so no versions are created; obsolete versions are lost. This state is the normal behavior of Swarm.	If you change the state from enabled to disabled, the health processor will erase the prior versions, which are now obsolete.
suspended	No new versions are accumulated but old versions are retained. This is a hybrid between enabled and disabled that preserves history.	If you re-enable versioning from this state, the chain of versions resumes from where it stopped.
enabled	New versions are accumulated as they are created, starting with any version that exists at the time versioning becomes enabled for the object.	

To implement versioning, you need to enable it by setting your versioning policy across all affected container objects, starting with the cluster:

- Cluster — set the configuration parameter `policy.versioning`. (This is a persisted setting, so set it via SNMP.)
- Domain(s) — set the domain object's `Policy-Versioning` and `Policy-Versioning-Unnamed` header values.
- Bucket(s) — set the bucket object's `Policy-Versioning` header value.

> Important
> Setting a header does not necessarily enable versioning. For example, you may have added `Policy-Versioning: enabled` to a bucket, but it has no effect if its domain does not have versioning explicitly enabled.

Status settings

The statuses for versioning vary by type of container object:

	Setting	Values	Description
Cluster	Config parameter: policy.versioning	disallowed suspended allowed	Status of versioning within the cluster. Defaults to disallowed. Important: This is a persisted setting, so be sure to set it by SNMP, not via configuration file: `snmpset -m +CARINGO-CASTOR-MIB -v2c -M +/usr/share/snmp/mib2c-data -cPASSWORD -OQs {swarm-ip} clusterConfig.policyVersioning s "allowed"`
Domain	Object header: Policy-Versioning	disabled suspended enabled required	Status of versioning for named objects within the domain, subject to the cluster setting. If the header is missing, the value is considered unspecified and versioning is disabled. Important To ensure that every bucket in a domain is versioned, set the domain's versioning state to **`required`**. Add a `Policy-Versioning: {value}` header using an SCSP PUT or COPY request on the domain object.
	Object header: Policy-Versioning-Unnamed	disabled suspended enabled	Status of versioning for unnamed objects that are tenanted in the domain, subject to the cluster setting. If the header is missing, the value is considered unspecified and versioning is disabled. Add a `Policy-Versioning-Unnamed: {value}` header using an SCSP PUT or COPY request on the domain object.
Bucket	Object header: Policy-Versioning	disabled suspended enabled	Status of versioning within the specific bucket, subject to the cluster and domain settings. If the header is missing, the value is considered unspecified a nd versioning is disabled. Important If the domain state is `enabled`, no bucket is versioned until it has its state changed to `enabled`. Add a `Policy-Versioning: {value}` header using an SCSP PUT or COPY request bucket object.

You can verify that versioning is enabled within a domain or bucket by making a HEAD request and checking the policy-related headers, `Policy-Versioning[-Unnamed]` (which you add) and `Policy-Versioning[-Unnamed]-Evaluated` (which Swarm generates). You control all `Policy-Versioning[-Unnamed]` headers, but Swarm computes and `Policy-Versioning[-Unnamed]-Evaluated` based on the policy of the complete context.

The versioning state of a domain depends on the cluster's state:

Domain Cluster	disabled	suspended	enabled	required (buckets only)
disallowed	disabled	disabled	disabled	disabled
suspended	disabled	suspended	suspended	suspended
allowed	disabled	suspended	enabled	enabled

Note
When the state is unspecified, it defaults to disabled. In a cluster that has versioning allowed, every newly created domain and bucket starts with an unspecified state, so object versioning is disabled until you enable it there explicitly.

The versioning state of a bucket depends on the parent domain's state:

Bucket Domain	disabled	suspended	enabled
disabled	disabled	disabled	disabled
suspended	disabled	suspended	suspended
enabled	disabled	suspended	enabled
required	enabled	enabled	enabled

Managing Polices across Clusters

If you are using replication feeds while using versioning, both the source and target clusters must run the same release of Swarm, but you choose how versioning occurs in the target:

Goal for Target	Target Cluster Policy	Effect
No versioning	(default) `policy.versioning = disallowed`	In the target cluster, Swarm maintains the current versions (no versioning) and cleans up any obsolete versions that are replicated. The domain and bucket objects replicate with the same policies, but the cluster-wide setting overrides them.

No versioning, access to versions	`policy.versioning = suspended`	In the target cluster, Swarm does not accumulate versions on local updates but preserves all of the prior versions that are replicated.
Identical versioning	`policy.versioning = allowed`	In the target cluster, Swarm performs versioning identically to the source cluster. The health processors in each cluster do the work of repairing the sequence of prior versions as they grow on both sides. Best practice To ensure faithful replication in your target cluster, be sure to change domain and bucket policies before you change the content they contain.
Custom versioning	`policy.versioning = allowed` and special syntax on custom policies	In the target cluster, Swarm performs versioning identically to the source cluster, except where you configured special context policies. Contact Support for the specific syntax needed to override policies at the domain or bucket level.

Keep in mind that policies are carried on contexts, and a remote cluster evaluates policies locally based on the contexts that it holds currently. This means that rapid changes to policies in a source cluster with changing content in those domains and buckets will give the correct results, but when those changes are replicated to the target cluster, they might play out differently.

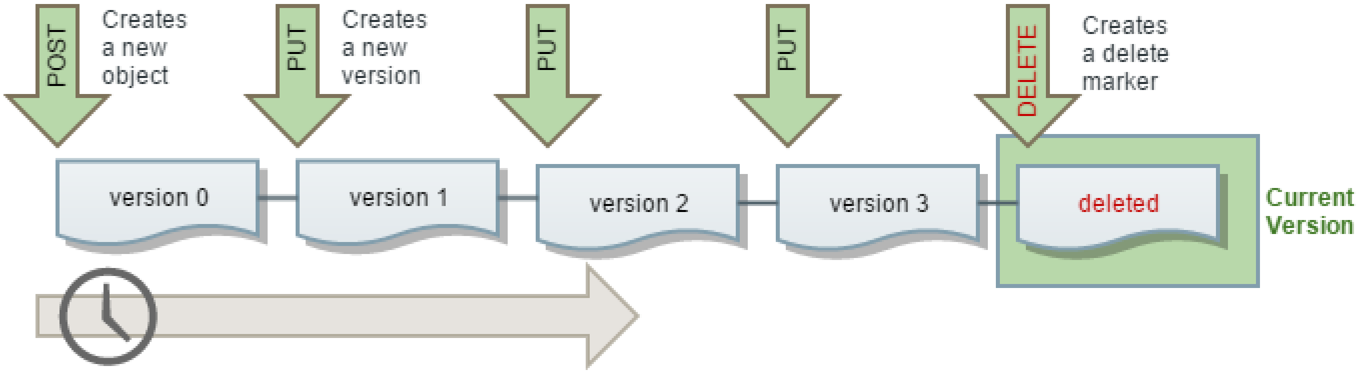
Working with Versioning

- When Versioning is Enabled
- When Versioning is Suspended

When Versioning is Enabled

When you enable or suspend versioning, the existing objects in your domain or bucket do not change: what changes is how Swarm handles future requests.

Once versioning is enabled in a context, Swarm uses each object's metadata to walk a virtual chain of versions of an object, from its creation to its deletion:



The order of the versions is the creation order, with the most recent one always being the current version.

These virtual version chains are extremely resilient: Swarm linearizes the linkage so that the newest version of an object links to the next in the ordering, which links to the next, and Swarms's health processing ensures that temporarily broken chains are repaired. Swarm orders all versions by the time they were written, regardless of the operations that created them.

> Note
>
> Only the sequence of the version chain is maintained: Swarm does not document change events, such as which operation caused the new version or that one is a restored version of another.

All of the usual SCSP operations act on the current version, which is the most recently updated one, unless a previous version is specifically mentioned on the request.

Key principles — These principles apply across all versioning operations, whether versioning is enabled or suspended:

- Regular operations always act on the current version of the object.
- Every version of each object — including a delete marker — is identified by its version identifier, which is its ETag value.
- Operations that include the version query argument always act on the specific version alone, even if it's the current version.

> Tip
>
> If you get a 404 Not Found or 400 Bad Request error when using the version query argument, this indicates that the context is versioning-disabled, which is the default if the versioning policy is still unspecified.

> See Versioning Operations and Versioning Examples.

When Versioning is Suspended

Suspending versioning lets you stop accruing new versions of the same object without jeopardizing the set of versions that already exists.

> Tip
>
> If you decide to undo version control altogether, you can disable versioning in the domain or bucket, which triggers the Health Processor to clean up all of the residual prior versions. This feature is not available in

Amazon S3.

When you suspend versioning, existing objects in your domain or bucket do not change: what changes is how Swarm handles future requests. You can re-enable versioning to have Swarm resume versioning behavior where it left off.

- New Objects — No versions are created for any objects created after versioning is suspended.
- Updating Objects — Swarm retains all existing versions when the versioning state is changed to suspended. After that, any update creates a new current version, which will be overwritten with updates after that.
- Retrieving Objects — Regardless of the versioning state of the domain or bucket, simple GET Object requests always return the current version of an object.
- Deleting Objects — If versioning is suspended, a DELETE request creates a delete marker for the object. You can also delete a specified version, which permanently removes that object.

Versioning Operations

- Adding Versions
- Retrieving Versions
- Listing Versions
- Deleting Versions
- Lifepoints for Versions
- Renaming Versioned Objects
- Restoring Versions

Note

If `version={etag}` is used where versioning is disabled, operations that reference the current version proceed normally, but any other ETag results in a 404 - Not Found. (v9.2)

Adding Versions

Any write operation (POST, PUT, COPY, APPEND) in a versioning-enabled context creates a new version, and it changes the existing version to be the prior one. (Prior versions are not modified. All write operations create new versions.)

- POST stores a brand new object as the current version. (PUT can be used if you have **`scsp.allowPutCreate`** enabled or add the **`putcreate`** query argument.)
- PUT adds a new version of the object, replacing the current version (which remains accessible by ETag); use PUT to update an existing alias. You cannot use PUT with the **`version`** query argument.
- COPY ?version={etag} creates a new version that duplicates the content of a versioned object with the new headers that you specify.
- APPEND ?version={etag} creates a new version that duplicates the headers of a versioned object with the new content that you specify (use 0-length to duplicate the existing content).

Retrieving Versions

Important

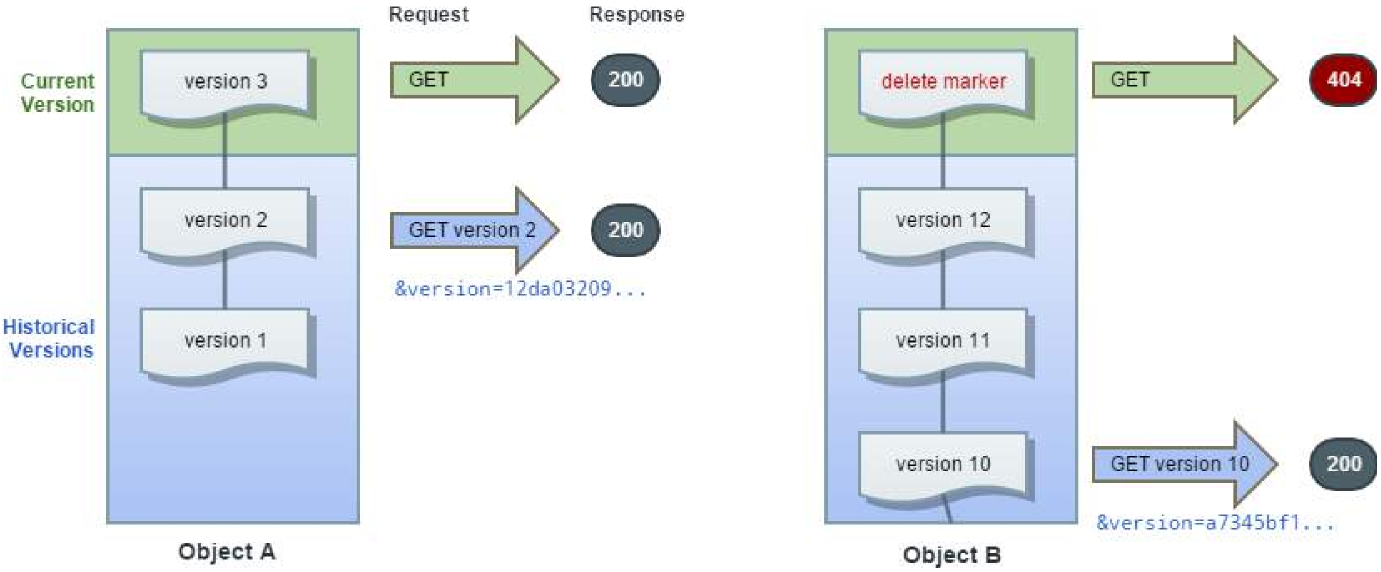
For the purpose of resource management, consider each version being added as being an entire object: it is not just a diff from the previous version. That is, if you have 12 versions of an object stored (the current and 11 prior ones), your object counts and space utilization will reflect 12 objects.

There are different ways to retrieve current objects versus specific object versions. To retrieve a specific version, you have to add the `version` query argument:

- GET retrieves the current version of an object.
- GET ?version={etag} retrieves a specific version of an object, identified by its ETag header value.
- HEAD retrieves the metadata of the current version of an object (and not its content).
- HEAD ?version={etag} retrieves the metadata of a specific version of an object, identified by its ETag.

To retrieve the content or metadata of a specific version, you have to specify the object's ETag, which retrieves the specified version of the object. This could be the current version or any one of the prior versions.

If you GET or HEAD a delete marker, you'll get a 404 Not Found response that includes the ETag for the delete marker only (not the prior version):



Listing Versions

With Elasticsearch indexing your cluster, you can use it to access listings of your object versions. (Elasticsearch is not required to use versioning.)

Versioning headers – Swarm returns these additional headers with historical (non-current) versioned objects:

- uuidNextVersion – Points to the primary UUID of the object that is the newer version. This field transfers during replication within the cluster but not to remote clusters (each cluster resolves its own chains).
- tmNextVersion – Holds the birth date of that newer (uuidNextVersion) object (they update as a pair). This field transfers during replication within the cluster but not to remote clusters.
- tmDeleted – Holds the death date of this particular version. This field does transfer to remote clusters so that the needed feed processing and clean up occurs.

> Note
>
> Unlike Amazon S3 (which retrieves only a maximum of 1000 objects), Swarm has no such limits, nor does it break batches on version boundaries.

To list all of the versions of all of the objects in a bucket, add the versions query argument to the GET Bucket request.

To list all of the version of one object, use prefix query arg along with versions query arg to the GET Bucket request, which limits the set of versions returned to only those related to that object.

```
GET /mybucket?domain=sample&format=json&versions HTTP/1.1
GET /mybucket?domain=sample&format=json&versions&prefix=objectName
HTTP/1.1
```

> Tip
>
> Add `&sort=tmborn:DESC,etag:DESC` to your listing operations to ensure that you are retrieving the versions in the precise order that Swarm is maintaining, starting with the current version. The secondary sort, ETag, is Swarm's tie-breaker for rapid updates in which two versions have the same tmborn.

Deleting Versions

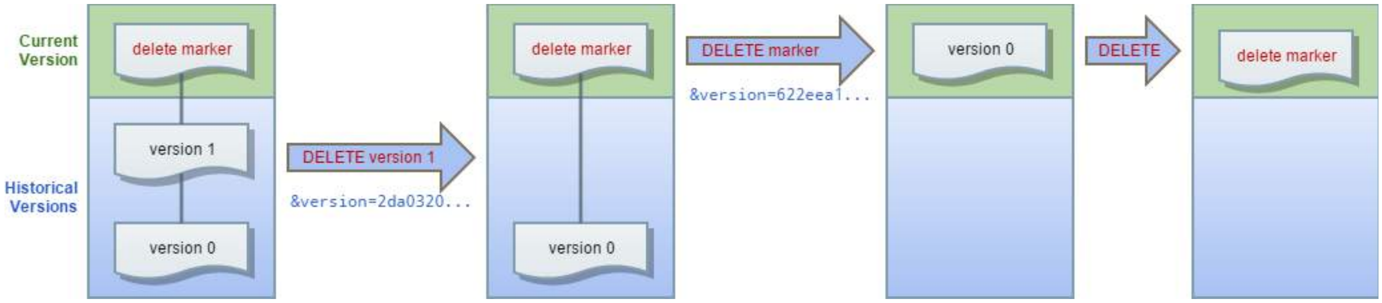
The behavior is the same when versioning is enabled or suspended:

- Deleting without specifying a version: Swarm operates on the current version and creates a delete maker.
- Deleting with a version specified: Swarm permanently pushes that version out of existence, even it’s the current version.

> Important
>
> When you delete a specific version of an object, it is removed permanently from the sequence of versions and from the cluster. You cannot recover these objects.

When versioning is enabled, a simple DELETE cannot permanently delete an object. Instead, Swarm inserts a delete marker, and that marker becomes the current version of the object with a new ID.

The response to a DELETE operation includes the ETag of the version being deleted, not that of the delete marker:



To permanently delete versioned objects, you must use DELETE Object and specify the ETag:

```
DELETE /mybucket/photo.gif?version=c347edc55b3c4fadb76d93022a29b07a
HTTP/1.1
```

Handling Delete Markers

A delete marker is a placeholder (marker) for a versioned object that was named in a simple DELETE request. Because the object was in a versioning-enabled bucket, the object was not deleted. The delete marker, however, makes Swarm behave as if it had been deleted. A delete marker has a name and version ETag like any other object, but it differs from other objects in several ways:

- It has no data.
- Its only operation is DELETE, which only the owner can request.
- It has nominal size.

Only Swarm can create a delete marker, and it does so whenever you send a DELETE Object request on an object that is versioning-enabled/suspended. The object named in the DELETE request is not actually deleted. Instead, the delete marker becomes the current version of the object. (The object's name becomes the name of the delete marker.)

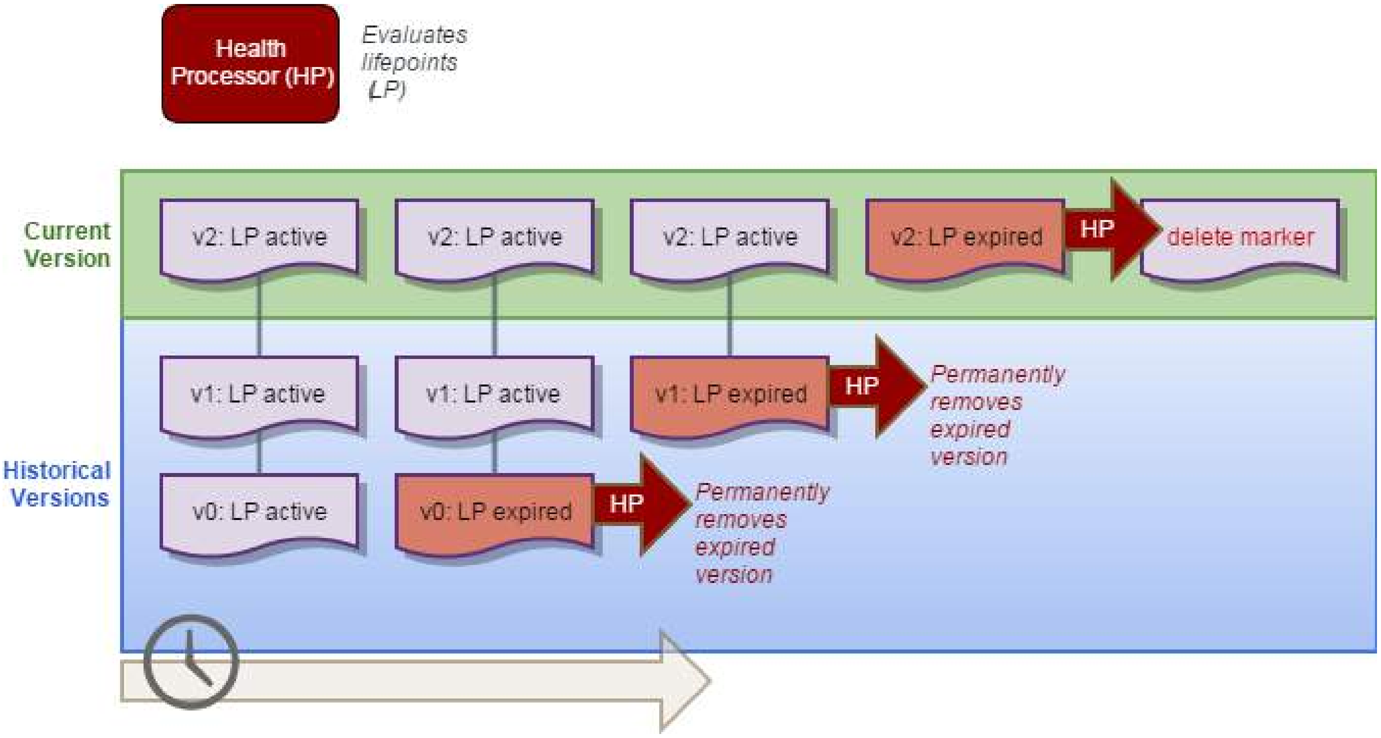
To permanently delete a delete marker, you must include its version ETag in a DELETE Object request. Only owners can permanently delete these markers while the context is versioning-enabled/suspended. (The health processor will clean up the markers along with the prior versions if the context returns to versioning-disabled.)

> Note
>
> The effect of removing the current delete marker is that the prior version of the object becomes the current version.

Lifepoints for Versions

You can delete object versions on demand, but you can also define lifepoints for objects that have a well-defined lifecycle to have Swarm permanently remove prior object versions as they expire. For both versioning-enabled and versioning-suspended states, the Health Processor cleans up (erases) versions with expired lifepoints, effectively deleting particular object versions. The Heath Processor then repairs the gap in the version sequence caused by the deletion.

When the lifepoint expires on the current version of the object, the Health Processor creates a delete marker that becomes the current version: it does not resurrect a previous version.



The Health Processor acts on versioned objects in these ways:

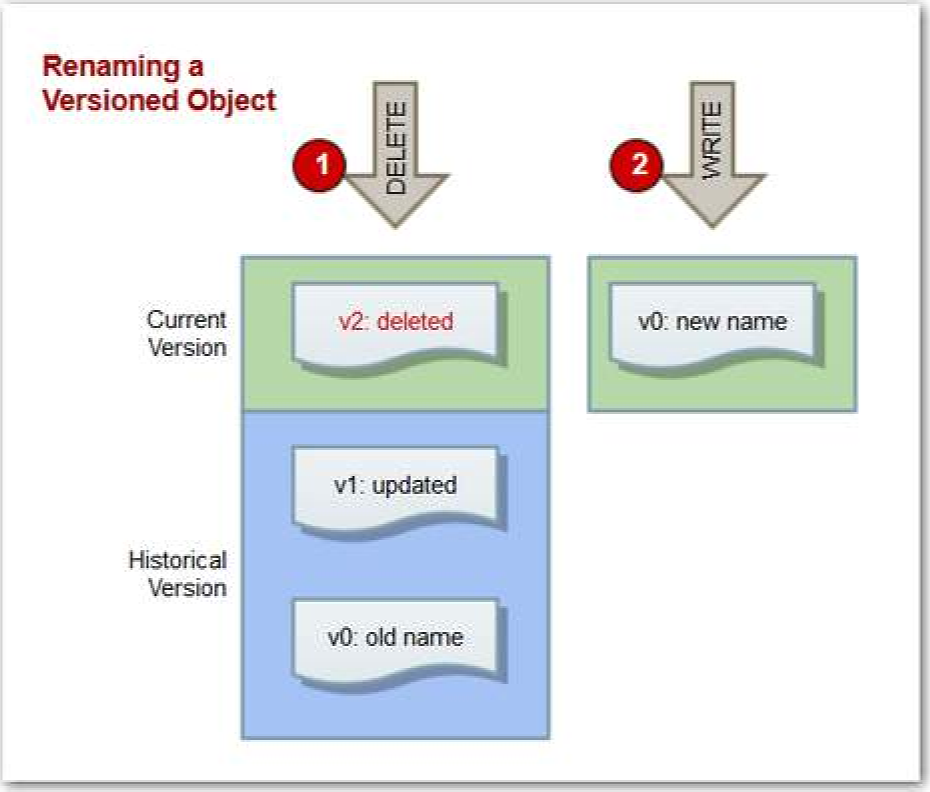
- Evaluates lifepoints of current versions, performing EC-related conversions or lifepoint deletions.
- Keeps the desired number of replicas of versioned objects.
- Maintains the current version of an object and the chain of version linkages.
- Deletes prior versions of objects that are no longer accessible.

> Note
> If the object is `versioning-enabled` or `versioning-suspended`, the Health Processor will not perform any erasure coding (EC) conversions, even on current versions.

Renaming Versioned Objects

Renaming a versioned object is done by ending the old one and starting a new one. That is, there are two requests and two objects:

1. a DELETE on the old name, which terminates the version chain with a delete marker
2. a WRITE of a new object, with the new name



Note that the name change can be reversed by deleting the new object, reading the last good version, and appending that version.

Restoring Versions

A key benefit of versioning is the ability to restore previous versions of an object. There are several ways to restore content by referencing specific versions, depending on your situation:

Problem	Method	Reason
The last update was erroneous	DELETE	Destroys: To undo an erroneous update that you don't want to save in the version history, permanently delete the current version by ETag: `version={etag}` When you delete the current object by its version ETag, Swarm automatically promotes the previous version to be the current version of that object.

An older version must be restored	APPEND	Preserves: To restore any one of the prior versions, copy a previous version of the object into the same context. The copied object becomes the current version of that object, and all object versions are preserved. APPEND adds data onto the end of an existing named object or aliased object while maintaining all of its populated metadata and object name or alias UUID. To duplicate the prior version exactly without change to content or metadata, use a 0-length APPEND. Because all object versions are preserved, you can make any earlier version the current version by copying a specific version of the object into the same bucket. Swarm supplies a new ID and it becomes the current version of the object (meaning a subsequent GET will retrieve the copy).
New metadata needs to be added	COPY	Replaces: To duplicate the prior version with updates to the metadata, use COPY. COPY updates the metadata on an existing object by copying its content verbatim while replacing its metadata. This technique makes it possible for you to extend your custom object metadata over time, to improve the scope and usefulness of your Elasticsearch queries.
An older version must be updated	~~PUT~~	Invalid: Prior versions are historical and cannot be altered (only deleted), so you cannot use PUT with them. Use COPY or APPEND on the older version, including the changes needed. If the existence of the incorrect version is a concern or a liability, use DELETE to remove the specific version permanently.

Versioning Examples

These examples show how you can use CURL commands for working with versioning.

- Enable versioning on a cluster
- Enable versioning on a domain
- Enable versioning on a bucket
- Create a named object in the bucket
- List versions in the bucket
- Update the named object
- View the new version in the listing
- INFO the prior version
- Delete the prior version
- Delete the current version
- Check the delete marker

Note

`<cluster>` in a URL stands for `<host>[:<port>]`, where `host` is a fully qualified domain name or IP address, plus a `port` number if other than 80. If the Host header does not match the domain name, override it with the domain= argument.

Enable versioning on a cluster

To allow versioning in a cluster, use an SNMP set command, adjusted for your environment:

```
snmpset -m +CARINGO-CASTOR-MIB -v2c -M +/usr/share/snmp/mib2c-data
-cPASSWORD -OQs {cluster} clusterConfig.policyVersioning s "allowed"
```

Enable versioning on a domain

To enable versioning for an existing domain, set its `Policy-Versioning` header and include the `admin` query argument:

Enable versioning on an existing domain via Swarm

```
curl --anyauth -u admin:password -i --location-trusted -X
PUT --post301 --data-binary ''
 -H "Policy-Versioning:
enabled" "{cluster}/?domain=sample&replicate=immediate&admin"

HTTP/1.1 201 Created
Location: {cluster}/?domain=sample
Volume: ec8ab948651deb7b9599d2288cf349e6
Location: {cluster}/?domain=sample
Volume: ae4adcf66f67f85ceb8096585fe8aa5e
Castor-System-Owner: @CAStor administrator
Entity-MD5: g9WWPqq3KoKxB5+dko3Taw==
Stored-Digest: 83d5963eaab72a82b1079f9d928dd36b
Last-Modified: Fri, 26 Jun 2015 21:52:07 GMT
Content-UUID: 157b02492dfb48d86fad8a0935ba440a
Castor-System-Version: 1435355527.814
Etag: "7f6d0be2bd765c68f41a960e8035e0f6"
Castor-System-Alias: 157b02492dfb48d86fad8a0935ba440a
Replica-Count: 2
Date: Fri, 26 Jun 2015 21:52:08 GMT
Server: CAStor Cluster/7.5.1
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400
```

Enable versioning on an existing domain via Gateway

```
curl --anyauth -u cs3admin:password -i --location-trusted -X PUT
--post301 --data-binary ''
 -H "Policy-Versioning:
enabled" "{cluster}/?domain=sample&replicate=immediate&admin"
 -v --anyauth
 -u cs3admin:caringo
 -H 'Content-Type: application/castorcontext'
```

To verify the versioning state for a domain, review the headers returned:

```
curl -i  --location-trusted --anyauth -u admin:ourpwdofchoicehere
 "{cluster}/?domain=sample&admin"

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="CAStor administrator",
nonce="3962288e3bb66797270d5a40d0d22468",
 opaque="f30a386668dae501437669a5572f12fe", stale=false, qop="auth",
algorithm=MD5
WWW-Authenticate: Basic realm="CAStor administrator"
Content-Length: 53
Content-Type: text/html
Date: Fri, 26 Jun 2015 21:52:22 GMT
Server: CAStor Cluster/7.5.1
Allow: HEAD, HOLD, GET, SEND, PUT, RELEASE, POST, COPY, GEN, APPEND,
DELETE
Keep-Alive: timeout=14400

HTTP/1.1 200 OK
Castor-System-Alias: 157b02492dfb48d86fad8a0935ba440a
Castor-System-CID: ffffffffffffffffffffffffffffffff
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 21:52:07 GMT
Castor-System-Name: sample
Castor-System-Owner: @CAStor administrator
Castor-System-Version: 1435355527.814
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
Last-Modified: Fri, 26 Jun 2015 21:52:07 GMT
Policy-Versioning: enabled
Etag: "7f6d0be2bd765c68f41a960e8035e0f6"
Castor-System-Path: /sample
Castor-System-Domain: sample
Volume: ae4adcf66f67f85ceb8096585fe8aa5e
Volume-Hint: ec8ab948651deb7b9599d2288cf349e6
Feed-0-Status: 0
Feed-0-StatusTime: Fri, 26 Jun 2015 21:52:09 GMT
Policy-Versioning-Evaluated: enabled
Date: Fri, 26 Jun 2015 21:52:22 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400
```

Notice that `Policy-Version-Evaluated` is `enabled`, which means that all of the tentanted alias objects in the domain will be versioned.

Enable versioning on a bucket

```
curl -i --location-trusted -X POST --post301 --data-binary '' -H
"Policy-Versioning: enabled"
 "{cluster}/mybucket?domain=sample&replicate=immediate"

HTTP/1.1 201 Created
Location: {cluster}/mybucket?domain=sample
Volume: 32745542cf1b9385aaf9b5a701544519
Location: {cluster}/mybucket?domain=sample
Volume: 56e4afcf1d25b43f8672d8f7fe7fbe59
Entity-MD5: U06twgE6BEijmq5+rM+MyA==
Stored-Digest: 534eadc2013a0448a39aae7eaccf8cc8
Last-Modified: Fri, 26 Jun 2015 21:53:27 GMT
Content-UUID: 779a221c6352785eafeef179d70a39d6
Castor-System-Version: 1435355607.358
Etag: "8e66a96c58508b0e256e6beb7c8009d5"
Castor-System-Alias: 779a221c6352785eafeef179d70a39d6
Replica-Count: 2
Date: Fri, 26 Jun 2015 21:53:27 GMT
Server: CAStor Cluster/7.5.1
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400

<html>
   <body>New object created</body>
</html>
```

Verify the versioning state of the bucket:

```
curl -i --location-trusted
 "{cluster}/mybucket?domain=sample"

HTTP/1.1 200 OK
Castor-System-Alias: 779a221c6352785eafeef179d70a39d6
Castor-System-CID: 157b02492dfb48d86fad8a0935ba440a
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 21:53:27 GMT
Castor-System-Name: mybucket
Castor-System-Version: 1435355607.358
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
Last-Modified: Fri, 26 Jun 2015 21:53:27 GMT
Policy-Versioning: enabled
Etag: "8e66a96c58508b0e256e6beb7c8009d5"
Castor-System-Path: /sample/mybucket
Castor-System-Domain: sample
Volume: 56e4afcf1d25b43f8672d8f7fe7fbe59
Policy-Versioning-Evaluated: enabled
Date: Fri, 26 Jun 2015 21:53:41 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400
```

Notice that `Policy-Version-Evaluated` is also `enabled`, meaning that all named objects in the bucket will be versioned.

Create a named object in the bucket

These two requests have typical headers.

```
curl -i --location-trusted -X POST --post301 --data-binary
'<html><body>First version's content.</body></html>' -H 'Content-type:
text/html'
 "{cluster}/mybucket/mydir/hello world.html?domain=sample"

HTTP/1.1 201 Created
Location: {cluster}/mybucket/mydir/hello%20world.html?domain=sample
Volume: 56e4afcf1d25b43f8672d8f7fe7fbe59
Last-Modified: Fri, 26 Jun 2015 22:02:35 GMT
Entity-MD5: VNvfqzhvwrgqWAl3Nc5q0w==
Stored-Digest: 54dbdfab386fc2b82a58097735ce6ad3
Castor-System-Version: 1435356155.156
Etag: "0f1d281546d24412c838058482eae868"
Date: Fri, 26 Jun 2015 22:02:35 GMT
Server: CAStor Cluster/7.5.1
Content-Length: 46
Content-Type: text/html
Keep-Alive: timeout=14400

<html>
   <body>New object created</body>
</html>
```

Verify the new object:

```
curl -i --location-trusted
 "{cluster}/mybucket/mydir/hello%20world.html?domain=sample"

HTTP/1.1 200 OK
Castor-System-CID: 779a221c6352785eafeef179d70a39d6
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 22:02:35 GMT
Castor-System-Name: mydir/hello%20world.html
Castor-System-Version: 1435356155.156
Content-Length: 39
Content-type: text/html
Last-Modified: Fri, 26 Jun 2015 22:02:35 GMT
Etag: "0f1d281546d24412c838058482eae868"
Castor-System-Path: /sample/mybucket/mydir/hello%20world.html
Castor-System-Domain: sample
Volume: 56e4afcf1d25b43f8672d8f7fe7fbe59
Date: Fri, 26 Jun 2015 22:02:43 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400

<html><body>First version's content.</body></html>
```

List versions in the bucket

You must have Elasticsearch enabled to use listing operations.

> Tip
>
> Always add `&sort=tmborn:DESC,etag:DESC` to your listing operations to retrieve the versions in the precise order that Swarm is maintaining, starting with the current version.

```
curl -i --location-trusted
 "{cluster}/mybucket?domain=sample&versions&sort=tmborn:DESC,etag:DESC"

HTTP/1.1 200 OK
Castor-System-Alias: 779a221c6352785eafeef179d70a39d6
Castor-System-CID: 157b02492dfb48d86fad8a0935ba440a
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 21:53:27 GMT
Castor-System-Name: mybucket
Castor-System-Version: 1435355607.358
Policy-Versioning: enabled
X-Timestamp: Fri, 26 Jun 2015 21:53:27 GMT
Last-Modified: Fri, 26 Jun 2015 22:04:40 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 1
Date: Fri, 26 Jun 2015 22:04:40 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400

[
   {
      "last_modified":"2015-06-26T22:02:35.156100Z",
      "hash":"0f1d281546d24412c838058482eae868",
      "content_type":"text/html",
      "name":"mydir/hello world.html",
      "bytes":39
   }
]
```

Update the named object

```
curl -i --location-trusted -X PUT --post301 --data-binary
'<html><body>Second version's content.</body></html>' -H 'Content-type:
text/html'
 "{cluster}/mybucket/mydir/hello world.html?domain=sample"

HTTP/1.1 201 Created
Location: {cluster}/mybucket/mydir/hello%20world.html?domain=sample
Volume: ec8ab948651deb7b9599d2288cf349e6
Last-Modified: Fri, 26 Jun 2015 22:09:26 GMT
Entity-MD5: 0rwew5/zuuYoeLNhLMJKKw==
Stored-Digest: d2bc1ec39ff3bae62878b3612cc24a2b
Castor-System-Version: 1435356566.913
Etag: "d953ada15686af402290cc77780249be"
Date: Fri, 26 Jun 2015 22:09:27 GMT
Server: CAStor Cluster/7.5.1
Content-Length: 54
Content-Type: text/html
Keep-Alive: timeout=14400

<html>
   <body>New object version created</body>
</html>
```

Verify the change to the named object:

```
curl -i --location-trusted
 "{cluster}/mybucket/mydir/hello%20world.html?domain=sample"

HTTP/1.1 200 OK
Castor-System-CID: 779a221c6352785eafeef179d70a39d6
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 22:09:26 GMT
Castor-System-Name: mydir/hello%20world.html
Castor-System-Version: 1435356566.913
Content-Length: 47
Content-type: text/html
Last-Modified: Fri, 26 Jun 2015 22:09:26 GMT
Etag: "d953ada15686af402290cc77780249be"
Castor-System-Path: /sample/mybucket/mydir/hello%20world.html
Castor-System-Domain: sample
Volume: ae4adcf66f67f85ceb8096585fe8aa5e
Date: Fri, 26 Jun 2015 22:13:20 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400

<html>
   <body>Second version's content.</body>
</html>
```

View the new version in the listing

```
curl -i --location-trusted
 "{cluster}/mybucket?domain=sample&versions"

HTTP/1.1 200 OK
Castor-System-Alias: 779a221c6352785eafeef179d70a39d6
Castor-System-CID: 157b02492dfb48d86fad8a0935ba440a
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 21:53:27 GMT
Castor-System-Name: mybucket
Castor-System-Version: 1435355607.358
Policy-Versioning: enabled
X-Timestamp: Fri, 26 Jun 2015 21:53:27 GMT
Last-Modified: Fri, 26 Jun 2015 22:14:23 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 2
Date: Fri, 26 Jun 2015 22:14:23 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400

[
   {
      "last_modified":"2015-06-26T22:09:26.913100Z",
      "hash":"d953ada15686af402290cc77780249be",
      "content_type":"text/html",
      "name":"mydir/hello world.html",
      "bytes":47
   },
   {
      "last_modified":"2015-06-26T22:02:35.156100Z",
      "hash":"0f1d281546d24412c838058482eae868",
      "content_type":"text/html",
      "name":"mydir/hello world.html",
      "bytes":39
   }
]
```

INFO the prior version

```
curl -i --location-trusted
 "{cluster}/mybucket/mydir/hello%20world.html?domain=sample&version=0f1d
281546d24412c838058482eae868"

HTTP/1.1 200 OK
Castor-System-CID: 779a221c6352785eafeef179d70a39d6
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 22:02:35 GMT
Castor-System-Name: mydir/hello%20world.html
Castor-System-Version: 1435356155.156
Content-Length: 39
Content-type: text/html
Last-Modified: Fri, 26 Jun 2015 22:02:35 GMT
Etag: "0f1d281546d24412c838058482eae868"
Castor-System-Path: /sample/mybucket/mydir/hello%20world.html
Castor-System-Domain: sample
Volume: 56e4afcf1d25b43f8672d8f7fe7fbe59
Date: Fri, 26 Jun 2015 22:05:22 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400
```

Delete the prior version

Deleting the historical version permanently erases it:

```
curl -i -X DELETE --location-trusted
 "{cluster}/mybucket/mydir/hello%20world.html?domain=sample&version=0f1d
281546d24412c838058482eae868"

HTTP/1.1 200 OK
Castor-System-Cluster: Sample
Content-Length: 0
Date: Fri, 26 Jun 2015 22:07:55 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400
```

You can list the versions again to verify the deletion:

```
curl -i --location-trusted
 "{cluster}/mybucket?domain=sample&sort=tmborn:DESC,etag:DESC"

HTTP/1.1 200 OK
Castor-System-Alias: 779a221c6352785eafeef179d70a39d6
Castor-System-CID: 157b02492dfb48d86fad8a0935ba440a
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 21:53:27 GMT
Castor-System-Name: mybucket
Castor-System-Version: 1435355607.358
Policy-Versioning: enabled
X-Timestamp: Fri, 26 Jun 2015 21:53:27 GMT
Last-Modified: Fri, 26 Jun 2015 22:24:52 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 1
Date: Fri, 26 Jun 2015 22:24:52 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400

[
   {
      "last_modified":"2015-06-26T22:09:26.913100Z",
      "hash":"d953ada15686af402290cc77780249be",
      "content_type":"text/html",
      "name":"mydir/hello world.html",
      "bytes":47
   }
]
```

Delete the current version

Deleting the current version adds a delete marker:

```
curl -i --location-trusted -X DELETE
 "{cluster}/mybucket/mydir/hello%20world.html?domain=sample"

HTTP/1.1 200 OK
Castor-System-CID: 779a221c6352785eafeef179d70a39d6
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 22:09:26 GMT
Castor-System-Name: mydir/hello%20world.html
Castor-System-Version: 1435356566.913
Content-type: text/html
Last-Modified: Fri, 26 Jun 2015 22:09:26 GMT
Etag: "d953ada15686af402290cc77780249be"
Content-Length: 0
Castor-System-Path: /sample/mybucket/mydir/hello%20world.html
Castor-System-Domain: sample
Volume: ae4adcf66f67f85ceb8096585fe8aa5e
Date: Fri, 26 Jun 2015 22:27:14 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400
```

Listing the versions shows the addition of a new version, the delete marker:

```
curl -i --location-trusted
 "{cluster}/mybucket?domain=sample&sort=tmborn:DESC,etag:DESC"

HTTP/1.1 200 OK
Castor-System-Alias: 779a221c6352785eafeef179d70a39d6
Castor-System-CID: 157b02492dfb48d86fad8a0935ba440a
Castor-System-Cluster: Sample
Castor-System-Created: Fri, 26 Jun 2015 21:53:27 GMT
Castor-System-Name: mybucket
Castor-System-Version: 1435355607.358
Policy-Versioning: enabled
X-Timestamp: Fri, 26 Jun 2015 21:53:27 GMT
Last-Modified: Fri, 26 Jun 2015 22:28:16 GMT
Transfer-Encoding: chunked
Content-Type: application/json; charset=utf-8
Castor-Object-Count: 2
Date: Fri, 26 Jun 2015 22:28:16 GMT
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400

[
   {
      "last_modified":"2015-06-26T22:27:14.643300Z",
      "hash":"86c3f94a9405067f10598b770cb00531",
      "content_type":"text/html",
      "name":"mydir/hello world.html",
      "bytes":0,
      "deletemarker":"true"
   },
   {
      "last_modified":"2015-06-26T22:09:26.913100Z",
      "hash":"d953ada15686af402290cc77780249be",
      "content_type":"text/html",
      "name":"mydir/hello world.html",
      "bytes":47
   }
]
```

Notice that the delete marker has the same name but is 0 bytes and has `"deletemarker":"true"`.

Check the delete marker

Getting INFO on the current version (the delete marker) results in a 404 Not Found error, but it also gives information about the delete marker:

```
curl -I --location-trusted
 "{cluster}/mybucket/mydir/hello%20world.html?domain=sample"

HTTP/1.1 404 Not Found
Castor-System-Name: mydir/hello%20world.html
Castor-System-Version: 1435357634.059
Content-Length: 0
Etag: "86c3f94a9405067f10598b770cb00531"
Date: Fri, 26 Jun 2015 22:35:43 GMT Be
Server: CAStor Cluster/7.5.1
Keep-Alive: timeout=14400
```

Note that the ETag (version ID) of the delete marker is returned.

Time of Last Access - atime

- Implementing atime
- Configuring atime
- Using atime with SCSP
- Using atime with Elasticsearch
- Limitations and Troubleshooting

Swarm can capture and persist the time of last access ("atime") on objects and add it to your search feed. This allows search queries to list objects that might be candidates for deletion or tiering (moving to cheaper storage). For example, you might write an application that uses atime values to purge "cold" objects that have not been read in the last three years. Swarm stores the atime as the `Castor-System-Accessed` header and indexes it as the `accessed` field in Elasticsearch, which is useful for bulk evaluations of content.

> Performance impacts
>
> Tracking atime does affect performance, so it should be enabled only if needed. Tracking access times can incur long-tail latencies on first reads, particularly when disk demands are heavy. For around 90% of objects that are read for the first time, the latency is negligible (<1 ms); only when requests queue on specific volumes do the effects become noticeable. Second reads have no performance impact. (SWAR-7772)
>
> Having high numbers of small object reads (such as thumbnail images) can cause memory indexes to run full.

Implementing atime

Because support for "atime" involves changes to the underlying Elasticsearch schema, you cannot restart an existing feed after Swarm upgrade, as the written and accessed fields would not be populated for some records and would have the incorrect type.

> Tip
>
> Although the atime feature works with Elasticsearch 2.3.3, it requires a rebuilding of your search index to use

the new schema, so take the opportunity to migrate to Elasticsearch 5.6 with the same reindexing.

1. For intensive READ access scenarios, provision additional memory to support the load on the in-memory index.
2. Finish installing the storage cluster to Swarm 10, and install the latest versions of the Swarm metrics and search RPMs in your Elasticsearch cluster.
3. Enable the cluster setting for the feature, which is disabled by default: `disk.atimeEnabled = true`
4. Create a new search feed, which will use the new Elasticsearch schema that supports atime.
5. If you had an existing search feed, complete these steps to transition to the new one:
   a. After the new feed completes processing, make the new feed the Primary.
   b. Pause the old feed.
   c. After verifying that the new feed is working as expected, you may delete the old feed and the old index data.

Configuring atime

The public settings for "atime" are dynamic, which means you can update these values on one node and Swarm will update all of the others, and the values persist across reboots. Following are all of the settings that control the gathering of atime information:

Settings for atime	Default	Type	Description
disk.atimeEnabled SNMP: accessedTimeEnabled	False	bool	Whether to track the time of last access on GET requests, stored in the Castor-System-Accessed header and indexed as the search field 'accessed'. Increases the load proportionally to the load of GETs in the cluster.
disk.atimeGranularity SNMP: accessedTimeGranularity	86400	int	In seconds; defaults to 1 day. The window of time during which atime will not be updated, which means that multiple reads may have occurred within that window of time. Lowering the value affects GET performance. A 1-second granularity gives most accurate accessed time results, but it results in a GET performance penalty due to increased disk access.
disk.atimeEnabledTime SNMP: accessedTimeEnabledTime	0	float	Non-UI. Read-only. The Linux epoch timestamp that records when `disk.atimeEnabled` was set to True. If the atime feature is later disabled, this time will be nulled out in SNMP, REST API, and phone home reports.

Using atime with SCSP

When you enable atime tracking for your cluster, Swarm keeps a record of the request time of each object's last write or read (successful GET request), and it sends that time to Elasticsearch as the accessed date field, for use in search queries. To access atime without Elasticsearch, you check the SCSP headers that Swarm adds to the objects.

With atime enabled, both SCSP HEAD and GET requests will include a Castor-System-Accessed header on the response whenever the verbose query argument is used. When requests return the Castor-System-Accessed response header, it has either the value of Castor-System-Created (because the object has not been read since the feature was enabled or the object was written) or else the read atime in the same GMT-based time format as Castor-System-Created. Keep in mind that the 1-day granularity (default) in updating atime means that additional reads may have occurred within that window of time.

Exceptions — GET requests trigger atime updates, except for these situations:

- Administrative and authorized admin requests
- Swarm requests for replication and other internal GET requests, such as for domains, settings, or manifests
- Any request with the special query argument to suppress recording atime: `notaccessed`
- Any request performing an integrity check or other specialized operation

> Tip
> The atime information is most useful on a HEAD request, because the atime on a GET request might be modified by the GET request itself, subject to the granularity.

To determine if an object has been read, HEAD the object using the `verbose` query argument.

If a read atime has not occurred, the Castor-System-Access value will match the Castor-System-Created:

```
> curl -I
http://192.168.1.12:80/5647f528ea85667a44dc754f975816c6?verbose
HTTP/1.1 200 OK
Castor-System-Alias: 5647f528ea85667a44dc754f975816c6
Castor-System-Cluster: Baker
Castor-System-Created: Wed, 19 Jul 2017 17:42:48 GMT
Castor-System-Accessed: Wed, 19 Jul 2017 17:42:48 GMT
...
```

If a read has occurred, the Castor-System-Access value is more recent than the Castor-System-Created:

```
> curl -I
http://192.168.1.12:80/5647f528ea85667a44dc754f975816c6?verbose
HTTP/1.1 200 OK
Castor-System-Alias: 5647f528ea85667a44dc754f975816c6
Castor-System-Cluster: Baker
Castor-System-Created: Wed, 19 Jul 2017 17:42:48 GMT
Castor-System-Accessed: Tue, 02 Oct 2018 23:03:56 GMT
...
```

Using atime with Elasticsearch

In Elasticsearch, the atime value is indexed as the accessed date field, which you can use in Swarm Search listing queries. Both the written and accessed fields will be populated in the Elasticsearch record:

Metadata Field	Type	Description
accessed	date (written and listed as ISO 8601)	If requested, the date of last access appears in listing results. The value does not reflect lifepoint conversion or segment consolidation that may have occurred. Matches the value for written until the first GET operation occurs, after which it updates for each qualified GET.
written	date (written and listed as ISO 8601)	Never changes for a particular object version (ETag).

Admin GET requests do not bump the atime value. You can also make SCSP GET requests with the notaccessed query argument, to suppress the atime update. This argument lets you list objects for management purposes without erroneously bumping the accessed date, as if an end-user or program had requested the object.

Argument	Value	Description
notaccessed	"yes"/"true"	If the atime feature is enabled, allows a GET request to complete without updating the accessed time on the object.

Limitations and Troubleshooting

- If the feature was enabled and disabled repeatedly, the reported value may be stale. To determine staleness, you can compare any value against the time when the atime feature was enabled: `atimeEnabledTime`.
- The reported atime, if present, is an atime that defaults to the granularity of a day; therefore, it is not necessarily the precise time (hours and minutes) of last access.
- Replication feeds do not get re-processed for atime changes.
- If a volume is lost, Swarm could lose a recent read atime for an object on that volume.
- If the atime feature is disabled after having been enabled, stale atimes remain in the Elasticsearch records. This means that applications using this field need to check the state of the feature (whether `disk.atimeEnabled = True`) and when the feature was last enabled (`disk.atimeEnabledTime`).
- For monitoring, Swarm provides a per-node count of how many objects have been assessed but whose atime has not yet been propagated to Elasticsearch.

Content Application Development

> See Concepts before you begin.

This section guides those in the following roles in developing applications that work with the Content Gateway:

- Storage system administrators
- End-user developers

The administrators are normally responsible for allocating storage, managing capacity, monitoring storage system health, replacing malfunctioning hardware, and adding additional capacity when needed. This can also include development staff responsible for automating storage administration functions.

End-user developers are responsible for creating custom application or integrating existing applications to use Content

Gateway storage.

- Gateway Metadata Transformation
- Metadata Translation between SCSP and S3
- Content Management API
- Content SCSP Extensions
- Token-Based Authentication
- Gateway Audit Logging
- Gateway Practical Applications
- Migrating Applications from Swarm Storage

Gateway Metadata Transformation

The metadata transformation facility allows domain and bucket administrators to define rules to add or replace metadata on incoming objects. These rules are stored in XFORM documents.

As a write operation (PUT, POST, or COPY) passes through the Gateway, rules in the xform sub-resource for a domain and/or bucket will be applied and the object's headers will be modified accordingly. Metadata rules specify that a given header will be added to the message if it does not exist or will be replaced if it does exist in the request. Headers defined in the domain XFORM take precedence over anything defined in a bucket XFORM.

Metadata Values

Metadata values are specified as strings, with a small number of variables available for substitution, using a ${varname } format where "varname" is the name of the variable.

For example, you could specify as a header value ${user}'s stuff on the user rooster's bucket and objects written into that bucket would end up with a header value of rooster's stuff.

Metadata Substitution Variables

Variable Name	Description
date:format	Create/update time stamp where format is defined by Java SimpleDateFormat specification
user	Authenticated user ID
domain	Domain name
bucket	Bucket name

XFORM Document Format

```
required Root := dict (
optional "metadata" := MetadataXforms
optional "comments" := any object type )
MetadataXforms := dict ( HttpHeaderName,HttpHeaderValue )
HttpHeaderName := str # Conforms to HTTP spec
HttpHeaderValue := str # Conforms to HTTP spec, plus variables
```

Example XFORM Document

```
{
   "comments": "Metadata transform document",
   "metadata": {
      "X-Written-When-Meta": "${date:yyyyMMdd-HHmmss}",
      "X-Contains-Meta": "${domain}/${bucket}",
      "X-Copyright-Meta": "Copyright ${date:yyyy}, MetaCorp, Inc",
      "X-Author-Meta": "${user}"
   }
}
```

Metadata Translation between SCSP and S3

As of release 5.4, Gateway performs translations of custom metadata formatting between the S3 and SCSP protocols, which means that it now provides S3 and SCSP applications the ability to access each other's metadata.

In order to allow SCSP and S3 clients to manipulate the full set of metadata that Swarm Storage supports, Gateway provides the following translations for Custom Metadata Headers:

SCSP		S3	
`x-*-meta`	🟦	`x-amz-meta-\1`	
`x-*-meta-*`	🟦	`x-amz-meta-\1-meta-\2`	
`x-amz-meta-*`	🟩	`x-amz-meta-\1`	one direction

Cyberduck Issues

If in Cyberduck you add two custom metadata values ("meta = cyberduck1" and "amz-meta = cyberduck2"), it's written as follows:

- x-amz-meta-meta: cyberduck1
- x-amz-meta-amz-meta: cyberduck2

which Gateway merges to the same header name.

SCSP HEAD

```
curl --head -u caringoadmin:password
'http://docker.example.com:9984/mybucket/duck.mpeg'
...
x-amz-storage-class-meta: STANDARD
x-amz-storage-class-meta: STANDARD
x-meta-meta: cyberduck1
x-meta-meta: cyberduck2
...
```

S3 HEAD

```
curl --head -u caringoadmin:password
'http://docker.example.com:9985/mybucket/duck.mpeg'
...
x-amz-meta-meta: cyberduck1
x-amz-meta-meta: cyberduck2
...
```

Cyberduck appears to use only the last header returned.

Content Management API

The Content Management API is an integration point for cloud management platforms and end-user applications. The Management API is purely administrative: it makes available actions for provisioning and managing storage tenants, domains, and other aspects of the Caringo cloud storage infrastructure using the same authentication and access control policy mechanism used within the Storage API.

The Management API is implemented as an HTTP/1.1 REST interface that is separate from the SCSP and S3 storage interfaces and is available for every user that can access the system. It works by overlaying the storage API name space at the /_admin/manage/ URI prefix.

> Note
> Since the _admin bucket is already a reserved resource for use by Caringo only, this name space overlay should have no effect on existing end-user applications.

- Namespace Structure
- Management API Response Formats

- Request Methods for Tenants
- Request Methods for Storage Domains
- Request Methods for Buckets
- Other Gateway Requests
- Defined ETC Documents
- Domain Adoption
- Methods for Quotas

Namespace Structure

The Content Management API namespace structure exists for every storage domain that is handled by the Gateway. It is a global URI mapping for all requests that come through the Gateway. The URI base for the Management API is:

URI base for Management API

```
/_admin/manage/
```

Below are the URI suffixes along with the HTTP methods and the corresponding Policy actions for each. If a Policy action is blank, the method is always allowed by non-authenticated requests.

> Note
> User-defined names supplied by the application, such as a tenant or domain name, are surrounded with curly braces, such as {tenant} or {domain}.

Management URI Methods and Policy Actions

URI Suffix	HTTP Methods	Policy Actions
Only Root Policy		
version	GET	
tenants	GET	ListTenants
meter/usage	GET	ListTenants
meter/status	GET	
tenants/{tenant}	PUT	CreateTenant
Merger of Root + Tenant Policy		
tenants/{tenant}	GET DELETE	GetTenant DeleteTenant

tenants/{tenant}/meter/usage	GET	GetTenant
tenants/{tenant}/etc	GET	ListEtc
tenants/{tenant}/etc/{document}	PUT GET DELETE	PutPolicy GetPolicy DeletePolicy
tenants/{tenant}/tokens	GET POST	ListTokens CreateToken
tenants/{tenant}/tokens/{token}	GET DELETE	ValidateToken DeleteToken
tenants/{tenant}/domains	GET	ListDomains
tenants/{tenant}/domains/{domain}	PUT (1)	CreateDomain
Merger of Root + Tenant + Domain Policy		
tenants/{tenant}/domains/{domain}	PUT (1) GET DELETE	PutDomain GetDomain DeleteDomain
tenants/{tenant}/domains/{domain}/meter/usage	GET	GetDomain
tenants/{tenant}/domains/{domain}/etc	GET	ListEtc
tenants/{tenant}/domains/{domain}/etc/{document}	PUT GET DELETE	PutPolicy GetPolicy DeletePolicy
tenants/{tenant}/domains/{domain}/uuid	GET	GetDomain
tenants/{tenant}/domains/{domainUUID}/name	GET	GetDomain
Merger of Root + Tenant + Domain + Bucket Policy		
tenants/{tenant}/domains/{domain}/buckets/{bucket}/uuid	GET	GetBucket
tenants/{tenant}/domains/{domain}/buckets/{bucketUUID}/name	GET	GetBucket

Note 1: The policy action for the PUT method on the `/_admin/manage/tenants/{tenant}/domains/{domain}` URI depends upon whether or not the storage domain already exists. If the domain is being created (does not exist),

CreateDomain can only be granted at the root or tenant scope and controls who can create a new domain. If the domain already exists, PutDomain controls who may change the domain and this can be granted at the root, tenant, or domain level.

Example: Getting the Management API version

```
GET /_admin/manage/version
Host: anydomain.cloud.example.com
```

The URI namespace table includes the appropriate Policy documents that are merged together when evaluating the access control policy for Management API requests. For example, to create a storage domain for a tenant, the Root and Tenant Policy documents are merged together. To manipulate a storage domain after it is already created, the Root, Tenant, and Domain Policy documents are all merged together.

System Tenant

Because the use of tenants is optional and because Swarm storage clusters may have existing storage domains created outside of Gateway, there is a concept called the SYSTEM TENANT that contains all of the storage domains in the cluster that are not assigned to a specific tenant. These are called untenanted storage domains and, for the purpose of API consistency, these storage domains are organized within a synthetic tenant named "`_system`" in the Management API.

Unlike other tenants, the system tenant does not have an owner, an IDSYS definition, a Policy, or authentication tokens. All domains within the system tenant are subject to the inheritance rules for the root IDSYS and Policy. These untenanted domains fall under the `/_admin/manage/tenants/_system/` URI path of the Management API.

Example: Listing untenanted storage domains

```
GET /_admin/manage/tenants/_system/domains/
Host: anydomain.cloud.example.com
```

Management API Response Formats

These are the Content Management API response formats that can be returned with a request. Unless otherwise noted, the order of the JSON fields in output records and the ordering of lists is not defined.

> Best practice
> For future compatibility, make your applications tolerant of extra fields in all responses.

- General Request Response
- Version Response
- Cluster Usage Response

General Request Response

This is the general-purpose response format for requests when no other specific format is defined. This response is also given when errors occur that prevent the return of a specific response format. For example, if a listing operation fails, this general request response format is used to communicate the error condition.

The response is a JSON formatted body whose fields and HTTP status codes are defined as follows:

General Request Response Format

message	human-readable message describing response
code	text response code
errors	array of strings with error details

Response Code Text Strings

WriteSucceeded	201 status code
DeleteSucceeded	200 status code
CreateFailed	various status codes from storage cluster
ListFailed	various status codes from storage cluster
ReadFailed	various status codes from storage cluster
WriteFailed	various status codes from storage cluster
DeleteFailed	various status codes from storage cluster
NotAuthorized	401 or 403 status code
ServerError	500 status code
MissingParameter	400 status code; missing required query argument
NotFound	404 status code; message contains path
OtherError	various status codes; non-specific error
BadJson	400 status code; JSON unable to be parsed

InvalidJson	400 status code; JSON didn't validate
BadRequest	400 status code; invalid argument or path

This is an example response:

```
{
  "message": "The data were not parsable JSON.",
  "code": "BadJson",
  "errors": [
    "Unexpected character ('P' (code 80)): expected a valid value
          (number, String, array, object, 'true', 'false' or 'null')\n
at
          [Source: [B@37994099; line: 1, column: 2]"
  ]
}
```

Version Response

The version response contains information about the Management API software version. A general request response is given if an error occurs on the request.

manageApiVersion	API version string; format not defined

This is an example version response:

```
{
  "manageApiVersion": "1.0"
}
```

Cluster Usage Response

The cluster usage response contains storage usage information about the back-end Swarm object storage cluster. A general request response is given if an error occurs on the request.

availableGb	Storage GBytes available; integer
capacityGb	Total storage GBytes in the cluster; integer

This is an example version response:

```
{
    "availableGb": 31676,
    "capacityGb": 512437
}
```

Tenant, Domain Listing Response

The tenant and domain listing response gives a listing of tenants and storage domains as a JSON list object. A general request response is given if an error occurs on the request.

name	Name of tenant or domain
etag	ETag opaque string; strong validator
contentMd5	base64 encoded MD5 value
lastModified	ISO 8601 format date and time with sub-second resolution

This is an example of a storage domain listing:

```
[
   {
      "name":"domain1.cloud.example.com",
      "etag":"74fd9321b793f0f62653e28f28e6e792",
      "contentMd5":"1B2M2Y8AsgTpgAmY7PhCfg==",
      "lastModified":"2013-11-24T20:06:07.719100Z"
   },
   {
      "name":"domain2.cloud.example.com",
      "etag":"237233ac1fddd2cd430920dcd133c3ac",
      "contentMd5":"1B2M2Y8AsgTpgAmY7PhCfg==",
      "lastModified":"2013-12-04T12:25:04.352100Z"
   },
   {
      "name":"domain3.cloud.example.com",
      "etag":"e224aa1d76ae7af7d77a94880f1a016e",
      "contentMd5":"1B2M2Y8AsgTpgAmY7PhCfg==",
      "lastModified":"2014-01-21T09:00:22.138100Z"
   }
]
```

ETC Listing Response

The etc listing response gives a listing of documents associated with a tenant or storage domain as a JSON list object. A general request response is given if an error occurs on the request.

Document Listing Response Format

name	Name of tenant or domain
etag	ETag opaque string; strong validator
contentMd5	base64 encoded MD5 value
lastModified	ISO 8601 format date and time with sub-second resolution

This is an example of a listing of etc documents:

```
[
   {
      "name":"policy.json",
      "etag":"786f68eed2afb0cd82ca325938a68ba1",
      "contentMd5":"SEkMcN3Q7wjtE2pek0tkYg==",
      "lastModified":"2014-01-08T22:15:23.725100Z"
   },
   {
      "name":"idsys.json",
      "etag":"3d5561edcecc6ea54d577fafcf0effc2",
      "contentMd5":"pYbOrt187VZtQzUsnATMQw==",
      "lastModified":"2014-01-08T22:05:44.826300Z"
   }
]
```

Token Response Formats

There are two types of token response formats: individual token GET/PUT response and token listing response. Both response formats are JSON documents. The token listing response is a JSON array of individual token objects.

owner	User name for whom this token applies
scope	Root (blank), tenant, or storage domain scope
token	Token identification; opaque value
secret	Optional S3 secret key; supplied during create

expiration	Token expiration in ISO 8601 format date and time with sub-second resolution
creation	Token creation in ISO 8601 format date and time with sub-second resolution

This is an example of a token response:

```
{
  "owner": "gcarlin",
  "scope": "",
  "token": "874fbb09057bc6be295fbdf4155deb73",
  "secret": "BaseballVsFootball",
  "expiration": "2016-02-05T04:05:55.000Z",
  "creation" : "2013-12-05T01:02:22.000Z"
 }
```

Token Listing Response Format

The format of the token listing response is a JSON array of token objects. By default, the token listing response does not include the secret fields for the tokens.

Request Methods for Tenants

This is detailed information about the request methods for tenants. Using these methods with curl has this format (here, for listing):

```
curl -i -u caringoadmin:pwd
https://site.example.com/_admin/manage/tenants/
```

Note
Tenant names are converted to lowercase before evaluation.

- List Tenants
- Create Tenant
- Read Tenant
- Delete Tenant
- List Tenant ETC Documents
- Create Tenant ETC Document
- Read Tenant ETC Document
- Delete Tenant ETC Document

- List Authentication Tokens
- Create Authentication Token
- Read Authentication Token
- Delete Authentication Token

> For more on ETC documents (IDSYS, Policy, XFORM), see Defined ETC Documents.

List Tenants

Method	GET
URI Suffix	tenants
Query Args	
Headers	
Policy Action	ListTenants
Description	Returns a list of tenants
Restrictions	Paging is not supported in the request; only 1000 returned at this time.
Request Body	
Response	JSON formatted tenant listing response

Create Tenant

Method	PUT
URI Suffix	tenants/{tenant}
Query Args	
Headers	Optional metadata to be saved with tenant
Policy Action	CreateTenant
Description	Create a tenant named {tenant}. If tenant already exists, this action overwrites the metadata for the tenant.
Restrictions	Tenant name must be 7-bit ASCII characters in the set [a-z, 0-9, hyphen].

Request Body	
Response	JSON general request response

Read Tenant

Method	GET
URI Suffix	tenants/{tenant}
Query Args	
Headers	
Policy Action	GetTenant
Description	Reads a tenant object and metadata
Restrictions	
Request Body	
Response	Tenant object body (normally null) and metadata

Delete Tenant

Method	DELETE
URI Suffix	tenants/{tenant}
Query Args	recursive=yes (required)
Headers	
Policy Action	DeleteTenant
Description	Deletes all data related to a tenant including storage domains
Restrictions	
Request Body	
Response	JSON general request response

List Tenant ETC Documents

Method	GET
URI Suffix	tenants/{tenant}/etc
Query Args	
Headers	
Policy Action	ListEtc
Description	Returns a list of tenant documents
Restrictions	Paging is not supported in the request; only 1000 returned at this time.
Request Body	
Response	JSON formatted documents listing response

Create Tenant ETC Document

Method	PUT
URI Suffix	tenants/{tenant}/etc/{document}
Query Args	
Headers	Any metadata to be included with the document
Policy Action	PutPolicy
Description	Create or overwrite a document associated with the tenant
Restrictions	Maximum document size is 1MB.
Request Body	The document contents
Response	JSON general request response

Read Tenant ETC Document

Method	GET

URI Suffix	tenants/{tenant}/etc/{document}
Query Args	
Headers	
Policy Action	GetPolicy
Description	Read a tenant document
Restrictions	
Request Body	
Response	Document body and metadata

Delete Tenant ETC Document

Method	DELETE
URI Suffix	tenants/{tenant}/etc/{document}
Query Args	
Headers	
Policy Action	DeletePolicy
Description	Delete a tenant document
Restrictions	
Request Body	
Response	JSON general request response

List Authentication Tokens

Method	GET
URI Suffix	tenants/{tenant}/tokens
Query Args	Set x-owner-meta={user} to search for another user's tokens; set withsecrets=true to include secret fields

Headers	
Policy Action	ListTokens
Description	List user authentication tokens
Restrictions	
Request Body	
Response	JSON formatted token listing response

Create Authentication Token

Method	POST
URI Suffix	tenants/{tenant}/tokens
Query Args	Set setcookie=true to have newly created token included in the Cookie response header. Set setcookie=false to prevent the newly created token from inclusion in a Cookie header in the response.
Headers	See Token-Based Authentication for details.
Policy Action	CreateToken
Description	Create user authentication token
Restrictions	Does not yet support creation of tokens for other users
Request Body	
Response	JSON formatted token response

Read Authentication Token

Method	GET
URI Suffix	tenants/{tenant}/tokens/{token}
Query Args	

Headers	
Policy Action	ValidateToken
Description	Read user authentication token
Restrictions	
Request Body	
Response	JSON formatted token response

Delete Authentication Token

Method	DELETE
URI Suffix	tenants/{tenant}/tokens/{token}
Query Args	
Headers	
Policy Action	DeleteToken
Description	Delete user authentication token
Restrictions	
Request Body	
Response	JSON general request response

Request Methods for Storage Domains

This is detailed information about the request methods for tenant storage domains. Using these methods with curl has this format (here, for listing):

```
curl -i -u caringoadmin:pwd
https://site.example.com/_admin/manage/tenants/t1/domains/
```

Note

Storage domain names are converted to lowercase before evaluation.

- List Tenant Domains
- Create Storage Domain
- Read Storage Domain
- Delete Storage Domain
- List Storage Domain ETC Documents
- Create Storage Domain ETC Document
- Read Storage Domain ETC Document
- Delete Storage Domain ETC Document
- Get Domain UUID by Name
- Get Domain Name by UUID

> For more on ETC documents (IDSYS, Policy, XFORM), see Defined ETC Documents.

List Tenant Domains

Method	GET
URI Suffix	tenants/{tenant}/domains
Query Args	
Headers	
Policy Action	ListDomains
Description	Returns a list of storage domains owned by the tenant.
Restrictions	
Request Body	
Response	JSON formatted tenant listing response

Create Storage Domain

Method	PUT
URI Suffix	tenants/{tenant}/domains/{domain}
Query Args	
Headers	Optional metadata to be saved with domain

Policy Action	CreateDomain
Description	Create a domain named {domain}. If domain already exists, this action overwrites the metadata for the domain.
Restrictions	Domain name must be 7-bit ASCII characters in the set [a-z, 0-9, hyphen].
Request Body	
Response	JSON general request response

Read Storage Domain

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}
Query Args	
Headers	
Policy Action	GetDomain
Description	Reads a storage domain object and metadata.
Restrictions	
Request Body	
Response	Domain object body (normally null) and metadata

Delete Storage Domain

Method	DELETE
URI Suffix	tenants/{tenant}/domains/{domain}
Query Args	recursive=yes (required)
Headers	
Policy Action	DeleteDomain

Description	Deletes all data related to the storage domain.
Restrictions	
Request Body	
Response	JSON general request response

List Storage Domain ETC Documents

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/etc
Query Args	
Headers	
Policy Action	ListEtc
Description	Returns a list of storage domain documents.
Restrictions	Paging is not supported in the request; only 1000 returned at this time.
Request Body	
Response	JSON formatted documents listing response

Create Storage Domain ETC Document

Method	PUT
URI Suffix	tenants/{tenant}/domains/{domain}/etc/{document}
Query Args	
Headers	Any metadata to be included with the document.
Policy Action	PutPolicy
Description	Create or overwrite a document associated with the storage domain.
Restrictions	Maximum document size is 1MB.

Request Body	The document contents
Response	JSON general request response

Read Storage Domain ETC Document

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/etc/{document}
Query Args	
Headers	
Policy Action	GetPolicy
Description	Read a storage domain document.
Restrictions	
Request Body	
Response	Document body and metadata

Delete Storage Domain ETC Document

Method	DELETE
URI Suffix	tenants/{tenant}/domains/{domain}/etc/{document}
Query Args	
Headers	
Policy Action	DeletePolicy
Description	Delete a storage domain document
Restrictions	
Request Body	
Response	JSON general request response

Get Domain UUID by Name

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/uuid
Query Args	
Headers	
Policy Action	GetDomain
Description	Gets the UUID (context ID) for a domain
Restrictions	
Request Body	
Response	JSON general request response

Get Domain Name by UUID

Method	GET
URI Suffix	tenants/{tenant}/domains/{domainUUID}/name
Query Args	
Headers	
Policy Action	GetDomain
Description	Gets the canonical name for a domain
Restrictions	
Request Body	
Response	JSON general request response

Request Methods for Buckets

This is detailed information about the request methods for buckets. Using these methods with curl has this format (here, for listing):

```
curl -i -u caringoadmin:pwd
https://site.example.com/_admin/manage/tenants/t1/domains/d1.site.exampl
e.com/buckets/
```

- List Buckets
- Create Bucket
- Read Bucket
- Delete Bucket
- List Bucket ETC Documents
- Create Bucket ETC Document
- Read Bucket ETC Document
- Delete Bucket ETC Document
- Get Bucket UUID by Name
- Get Bucket Name by UUID

> For more on ETC documents (IDSYS, Policy, XFORM), see Defined ETC Documents.

List Buckets

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/buckets
Query Args	
Headers	
Policy Action	ListBuckets
Description	Returns a list of buckets in the domain
Restrictions	
Request Body	
Response	JSON-formatted listing response

Create Bucket

Method	PUT

URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}
Query Args	
Headers	Optional metadata to be saved with bucket
Policy Action	CreateBucket
Description	Create a bucket named {bucket}. If it already exists, this action overwrites the metadata for the bucket.
Restrictions	Name must be 7-bit ASCII characters in the set [a-z, 0-9, hyphen]
Request Body	
Response	JSON general request response

Read Bucket

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}
Query Args	
Headers	
Policy Action	GetBucket
Description	Reads a bucket object and metadata.
Restrictions	
Request Body	
Response	Bucket object body (normally null) and metadata

Delete Bucket

Method	DELETE
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}

Query Args	recursive=yes (required)
Headers	
Policy Action	DeleteBucket
Description	Deletes all data related to the bucket
Restrictions	
Request Body	
Response	JSON general request response

List Bucket ETC Documents

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}/etc
Query Args	
Headers	
Policy Action	ListEtc
Description	Returns a list of bucket documents
Restrictions	Paging is not supported in the request; only 1000 returned
Request Body	
Response	JSON-formatted documents listing response

Create Bucket ETC Document

Method	PUT
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}/etc/{document}
Query Args	
Headers	Any metadata to be included with the document

Policy Action	PutPolicy
Description	Create or overwrite a document associated with the bucket
Restrictions	Maximum document size is 1MB
Request Body	The document contents
Response	JSON general request response

Read Bucket ETC Document

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}/etc/{document}
Query Args	
Headers	
Policy Action	GetPolicy
Description	Read a bucket document
Restrictions	
Request Body	
Response	Document body and metadata

Delete Bucket ETC Document

Method	DELETE
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}/etc/{document}
Query Args	
Headers	
Policy Action	DeletePolicy
Description	Delete a bucket document

Restrictions	
Request Body	
Response	JSON general request response

Get Bucket UUID by Name

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucket}/uuid
Query Args	
Headers	
Policy Action	GetBucket
Description	Gets the UUID (context ID) for a bucket
Restrictions	
Request Body	
Response	JSON general request response

Get Bucket Name by UUID

Method	GET
URI Suffix	tenants/{tenant}/domains/{domain}/buckets/{bucketUUID}/name
Query Args	
Headers	
Policy Action	GetBucket
Description	Gets the canonical name for a bucket
Restrictions	
Request Body	

Response	JSON general request response

Other Gateway Requests

This is detailed information about the request methods for other Content Management API resources.

- Read API Version
- Read Cluster Storage Usage

Read API Version

Method	GET
URI Suffix	version
Query Args	
Headers	
Policy Action	
Description	Read the Management API version information
Restrictions	
Request Body	
Response	JSON version response

Read Cluster Storage Usage

Method	GET
URI Suffix	meter/cluster/usage
Query Args	
Headers	
Policy Action	ListTenants
Description	Read the storage usage information for the back-end cluster
Restrictions	

Request Body	
Response	JSON cluster usage response

Defined ETC Documents

The Content Gateway makes use of the etc document storage for tenants and storage domains in order to store IDSYS, Policy, and XFORM information. These defined document names are used by the Gateway and are exposed through the Management API as an end-point for integration with applications.

- IDSYS
- Policy
- XFORM

IDSYS

The IDSYS documents for tenants and storage domains are created and modified by uploading the JSON document through the Management API.

```
tenants/{tenant}/etc/idsys.json
tenants/{tenant}/domains/{domain}/etc/idsys.json
```

The entire JSON document with all fields must be provided when updating the idsys.json document and the Content-Type: application/json header must be included with the request.

- Permission to create and update is granted with the PutPolicy policy action.
- Reading the IDSYS document is controlled with the GetPolicy policy action.

The storage domain's IDSYS can also be manipulated through the SCSP Storage API.

Policy

The Policy documents for tenants and storage domains are created and modified by uploading the JSON document through the Management API.

```
tenants/{tenant}/etc/policy.json
tenants/{tenant}/domains/{domain}/etc/policy.json
```

The entire JSON document with all fields must be provided when updating the policy.json document and the Content-Type: application/json header must be included with the request.

- Permission to create and update is granted with the PutPolicy policy action.
- Reading the Policy document is controlled with the GetPolicy policy action.

The access control policies for domains and buckets can also by manipulated through the SCSP Storage API.

XFORM

The metadata transform (XFORM) document for storage domains is created and modified by uploading the JSON document through the Management API.

```
tenants/{tenant}/domains/{domain}/etc/metaxform.json
```

The entire JSON document with all fields must be provided when updating the metaxform.json document and the Content-Type: application/json header must be included with the request.

- Permission to create and update is granted with the PutPolicy policy action.
- Reading the XFORM document is controlled with the GetPolicy policy action.

The metadata transform can also be manipulated through the SCSP Storage API.

Domain Adoption

If you have a storage domain that was created outside the Content Management API (such as a replication cluster), it is possible to have a tenant adopt it so that you can access it through the Content Portal. However, if that storage domain was created with Swarm's legacy auth/auth, there is a special variation to the domain adoption procedure, noted below.

For domain adoption, ensure that:

- The tenant exists.
- The cluster administrator executes these actions directly against the storage cluster and not through the Gateway.

> Note
> This is a highly privileged operation that has no equivalent request within the Gateway.

To adopt a domain

1. Get all of the custom metadata that is attached to the domain.
2. Update the domain providing all custom metadata and an `x-tenant-meta-name` header.

This is an example of the commands. The strings {tenant} and {domain} are substituted for the actual tenant name and storage domain name.

The first step is to retrieve all of the current, custom metadata name/value pairs for the domain:

```
HEAD /?domain={domain}
```

If the domain has Swarm's legacy auth/auth on it, there will be a `Castor-Authorization` header in the response or you may receive a "401 Unauthorized" response to the previous request. See the section below for instructions to remove the legacy auth/auth.

> Deprecated
> The native Swarm auth/auth feature is deprecated and will be removed after June 2017.

Only certain field names are valid as custom metadata. In general, you can use `Castor-*` (except for `Castor-System-*`), `Content-*`, `X-*-Meta`, and `X-*-Meta-*` headers in the HEAD response as custom metadata for the storage domain. With the exception of `Castor-Authorization`, these are the fields that you will want to preserve.

> For details about headers, see SCSP Headers and SCSP COPY.

After getting all of the custom metadata name/value pairs (denoted as {mdName#} and {mdValue#}) from the HEAD request, use the COPY request to replace all of the storage domain object's metadata and include the adoptive tenant's name:

```
COPY /?domain={domain}&replicate=immediate
    x-tenant-meta-name: {tenant}
    {mdName1}: {mdValue1}
    {mdName2}: {mdValue2}
    ...
```

The `x-tenant-meta-name` must match the name of an existing tenant that was created through the Gateway Management API or Content Portal.

Upon completion of the domain adoption procedure, the storage domain will now be subject to the tenant access control policy in addition to the root and domain policies. Additionally, if the storage domain does not define its own IDSYS, was previously using the root IDSYS, and the adoptive tenant defines an IDSYS, the storage domain will switch to using the tenant IDSYS instead of the root IDSYS.

Removing legacy auth/auth

If the storage domain has the legacy auth/auth on it, you need to remove it so that the domain can be used correctly through Gateway. These examples will use the curl command line utility since it is able to perform HTTP digest authentication. You can adapt these examples for use with another tool or library in order to issue the HEAD and COPY commands. Since this process is very similar to the previous one for domains without legacy auth/auth, references will be made to the instructions from the previous section. The {adminUser} username in these examples must be for one of the Swarm administrators defined in the storage cluster's configuration. The {storageNode} string is the host or IP for any node in the storage cluster.

Get the current metadata for the storage domain.

```
curl -I --digest -u {adminUser}
 --location-trusted
 'http://{storageNode}/?domain={domain}'
```

You need all of the custom metadata name/value pairs with the exception of the `Castor-Authorization` header.

The previously described COPY request will be performed using HTTP digest authentication.

```
curl -X COPY --digest -u {adminUser}
 --location-trusted
 -H 'x-tenant-meta-name: {tenant}'
 -H '{mdName1}: {mdValue1}'
 -H '{mdName2}: {mdValue2}'
 ...
 'http://{storageNode}/?domain={domain}&replicate=immediate'
```

All of the prior discussion about the `x-tenant-meta-name` value and post-creation domain behavior apply.

Methods for Quotas

In addition to specifying quota policies directly in the Content Portal, you can set and clear quotas and check on quota statuses using the Content Management API:

> See Setting Quotas and Content Metering.

Legend:

- {M} = metric name, one of “bandwidth”, “rawstorage”, “storage”
- {T} = tenant name
- {D} = domain name
- {B} = bucket name

Method and Suffix	Query Arguments	Policy Action

PUT /_admin/manage/tenants/{T}/
• quota/{M}/limit
• domains/{D}/quota/{M}/limit
• domains/{D}/buckets/{B}/quota/{M}/limit
	limit={integer}{KB|MB|GB|TB}
	state={ok|notify|nowrite|read|lock}
	None removes current values
		PutQuo

PUT /_admin/manage/tenants/{T}/
• quota/{M}/override
• domains/{D}/quota/{M}/override
• domains/{D}/buckets/{B}/quota/{M}/override
	duration={number}{s|m|d|w}
	deadline={timestamp}
	statduration={ok|notify|nowrite|read|lock}
	None removes current values
		PutQuo

PUT /_admin/manage/tenants/{T}/ • quota/email • domains/{D}/quota/email • domains/{D}/buckets/{B}/quota/email	addresses={email}[,{email},...] None removes current values	PutQuot
HEAD /_admin/manage/tenants/{T}/ • quota/{M} • domains/{D}/quota/{M} • domains/{D}/buckets/{B}/quota/{M} • quota/* • domains/{D}/quota/* • domains/{D}/buckets/{B}/quota/*	-	GetQuot

HEAD /_admin/manage/tenants/{T}/ • quota/check • domains/{D}/quota/check • domains/{D}/buckets/{B}/quota/check	-	GetQuot
GET /_admin/manage/quota/status	-	ListDom

GET /_admin/manage/tenants/{T}/
• domains/{D}/meter/usage/bytesIn
• domains/{D}/meter/usage/bytesOut
	from={timestamp}
	to={timestamp}	

GET /_admin/manage/tenants/{T}/ • meter/usage/bytesSize/current • domains/{D}/meter/usage/bytesSize/current • domains/{D}/buckets/{B}/meter/usage/bytesSize/current • meter/usage/bytesStored/current • domains/{D}/meter/usage/bytesStored/current • domains/{D}/buckets/{B}/meter/usage/bytesStored/current	-	

Content SCSP Extensions

This section documents the Content Gateway enhancements to the Swarm SCSP client protocol.

> Note
> These SCSP protocol changes are only applicable when communicating to the object storage cluster through the Gateway.

- SCSP Context Sub-resources
- Domain and Bucket Creation

- Recursive Deletes
- Multipart MIME POST
- Gateway ACL for Objects
- Gateway CORS for Buckets

SCSP Context Sub-resources

The Gateway creates SCSP context sub-resources in order to allow the specification of identity management systems, access control policies, and metadata transforms.

These are the sub-resources and the context in which they are applicable when using the Gateway.

Sub-resource	Context	Description
idsys	domain	Identity system definition
policy	domain, bucket	Access control policy
xform	domain, bucket	Metadata transform

All storage domain and bucket sub-resources are controlled with one of the policy actions PutPolicy, GetPolicy, or DeletePolicy.

> Warning
> Permission to read or change these sub-resources for a storage domain must be protected from untrusted users and, in deployments where end-users are allowed to manage their storage domains, a cluster or tenant administrator will normally retain ownership of the storage domain. If an end-user owns the storage domain, they will be able to read and change the domain's sub-resources.

- IDSYS
- Policy
- XFORM

IDSYS

The IDSYS document sub-resource for a storage domain is manipulated using authenticated SCSP commands through the Gateway. This is accomplished by uploading the JSON document for the IDSYS to the storage domain's idsys sub-resource using the HTTP PUT operation.

```
PUT /?idsys Content-Type: application/json
{"ldap" : {
"ldaphost" : "ldap.example.com", ...
}
```

The entire JSON document with all fields must be provided when updating the idsys sub-resource and the Content-Type: application/json header must be included with the request.

Permission to update the IDSYS document for a domain is granted with the PutPolicy policy action.
Reading the IDSYS document is controlled with the GetPolicy policy action and uses the HTTP GET operation.

```
GET /?idsys
```

Similarly, an IDSYS is removed using the HTTP DELETE operation and controlled with the DeletePolicy policy action.

```
DELETE /?idsys
```

Policy

The Policy document sub-resources for storage domains and buckets are manipulated using authenticated SCSP commands through the Gateway.

Creating a new Policy document or replacing an existing one are both controlled with the PutPolicy action. The entire JSON document with all fields must be provided when updating the policy sub-resource and the Content-Type: application/json header must be included with the request.

The HTTP PUT operation is used to update a domain Policy:

```
PUT /?policy
 Content-Type: application/json
 {"Id":"My Domain Policy", ... }
```

...or a bucket Policy:

```
PUT /mybucket?policy
 Content-Type: application/json
 {"Id":"My Bucket Policy", ...}
```

Reading a Policy document is controlled by the GetPolicy action. Examples of reading a Policy for a storage domain and a bucket:

```
GET /?policy
GET /mybucket?policy
```

Deleting a Policy document is controlled by the DeletePolicy action. Examples of deleting a Policy for a storage domain and a bucket:

```
DELETE /?policy
DELETE /mybucket?policy
```

XFORM

The metadata transform (XFORM) sub-resource for domains and buckets are manipulated using authenticated SCSP commands through the Gateway.

Creating a new XFORM document or replacing an existing one are both controlled with the PutPolicy action. The entire JSON document with all fields must be provided when updating the xform sub-resource and the Content-Type: application/json header must be included with the request.

The HTTP PUT operation is used to update a domain XFORM:

```
PUT /?xform
{"metadata" : { ... }}
```

Or a bucket XFORM:

```
PUT /mybucket?xform
{"metadata" : { ... }}
```

Reading an XFORM document is controlled by the GetPolicy action. Examples of reading an XFORM for a storage domain and a bucket:

```
GET /?xform
GET /mybucket?xform
```

Deleting an XFORM document is controlled by the DeletePolicy action. Examples of deleting an XFORM for a storage domain and a bucket:

```
DELETE /?xform
DELETE /mybucket?xform
```

Domain and Bucket Creation

Swarm Storage has two types of context objects: domains and buckets. When you create them from Gateway, follow these guidelines:

Required Header

When clients create these contexts, Gateway requires that the HTTP POST request include the following header to identify the content type as a context object:

```
Content-Type: application/castorcontext
```

If contexts are not explicitly identified as shown during creation, the Gateway will return an HTTP 400 Bad Request error response to the client.

> Note
> Contexts that already exist in the storage cluster or that are created directly to the cluster do not require this `Content-Type` specification in order to work properly with Gateway.

Required Argument

SCSP requires you to add the `domain` query argument when creating a storage domain, regardless of the existence of a `X-Forwarded-Host` or `Host` header on the request.

Optional Sub-resources

Gateway supports the common application need to immediately create one or more of the IDSYS, Policy, and XFORM sub-resources for the new context in a one-shot request. This is done by including the additional documents within a JSON formatted request body on the context create request. If you do not need this one-shot creation of authentication for the context, the context is created with an HTTP POST request that is submitted with a null request body, `Content-Length` zero, and `Content-Type` of `application/castorcontext`.

The format of the JSON body of a one-shot creation request must conform to the following JSON schema.

> For information about JSON schemas, see http://json-schema.org.

```
{
  "type": "object",
  "properties": {
    "config": {
      "type": "object",
      "properties": {
        "idsys": {
          "$ref":
"https://support.cloud.caringo.com/schemas/idsys-schema.json#"
        },
        "policy": {
          "$ref":
"https://support.cloud.caringo.com/schemas/policy-schema.json#"
        },
        "metaxform": {
          "$ref":
"https://support.cloud.caringo.com/schemas/metaxform-schema.json#"
        }
      },
      "optional": true,
      "additionalProperties": false
    },
    "metadata": {
      "type": "object",
      "optional": true
    }
  },
  "additionalProperties": false
}
```

This is an example JSON document that uses the schema:

```
{
  "config": {
    "idsys": { ... },
    "policy": { ... },
    "metaxform": { ... }
  },
  "metadata": {
    "X-Copyright-Meta": "Copyright 2015 Widgets, Inc."
  }
}
```

Any combination of the IDSYS, Policy, and XFORM document sub-resources can be included in the one-shot request. Keep in mind that IDSYS is only valid for a storage domain context and not for bucket contexts. The same configuration document formats described in this guide are used within the "{ ... }" portions of the example.

After a context object has been created, updates to its IDSYS, Policy, or XFORM sub-resources must be done individually and cannot use the composite form of the one-shot creation request.

Recursive Deletes

Gateway utilizes Swarm's recursive deleting mechanism with some augmentation for requests that do not contain a "`recursive={value}`" query argument.

> For more about recursive deletes of storage domains and buckets, see Managing Domains and SCSP Query Arguments.

When removing an empty storage domain or an empty bucket, if the client request does not include the recursive query argument, Gateway will automatically add "`recursive=yes`" to the DELETE request that is transmitted to Swarm. This preserves the protocol behavior of the Gateway versions prior to 4.0. This is analogous to the Unix `rmdir` behavior on empty directories.

If a client issues a DELETE on a storage domain or bucket that is not empty, the request must include the recursive query argument or else the request will fail. This is analogous to attempting to run `rmdir` on a non-empty directory in Unix.

The Gateway client audit log will record the single DELETE operation. Even if the delete is permitted and the domain or bucket contains objects, the individual objects will not receive individual audit log records of their deletion.

> Although Swarm's recursive deletes are always asynchronous, some deletes may be subject to a delay period. See SCSP Query Arguments for explanation of the "`recursive=yes`" and "`recursive=now`" delete options.

> Important
>
> Content Portal uses the "`recursive=now`" delete option for storage domain and bucket removal.

Multipart MIME POST

Content Gateway allows client applications to use the HTTP multipart MIME POST in order to upload multiple files in one operation. In turn, Gateway converts these multiple parts into individual POST operations to Swarm. While Gateway always returns an HTTP 202 response code, the body of the response contains the results of the individual POST operations. The Content Portal uses this mechanism for the upload page.

While processing this type of request, the individual files are extracted from the original POST request and spooled to the Gateway server's local file system before transmitting them to Swarm. The spool directory is specified with the `multipartSpoolDir` setting and is allowed to fill the file system up to a maximum percentage defined with the `multipartUsageAllowed` setting.

> Important
>
> You must ensure that there is sufficient available disk space in the file system in order to handle the incoming requests.

The multipart MIME POST request is the only type of request that uses a local disk spool on the Gateway. Requests such as SCSP single-object writes, S3 multipart uploads, and SCSP multipart writes are all streamed directly to Swarm.

Gateway ACL for Objects

- GET Object ACL
- GET Object Version ACL
- PUT Object ACL
- PUT Object Version ACL

Gateway supports management of access control lists (ACLs) for objects, including changes to existing authorizations.

> Differences from S3
>
> - `PUT /object?acl&versionId=X` cannot be used because metadata on old versions is immutable.
> - Updating an object’s acl also updates the object’s modification time.
> - Gateway uses a convention whereby user names are decorated with `{username}@{domain}` or `{username}+{tenant}` depending on the idsys in which the user is defined. If the user is defined in the root idsys, then the decoration looks like `{username}@`. This decoration may be omitted if there cannot be any ambiguity, but, internally, Gateway will always store decorated usernames in the ACL owner and user grantees, adding it to an incoming ACL as needed and removing it where possible before passing an ACL back to the client.

GET Object ACL

GET Object acl uses the acl subresource to return the access control list (ACL) of an object. To use this operation with S3, you must have READ_ACP access to the object.

The following request returns information, including the ACL, of an object:

```
GET /{object-name}?acl HTTP/1.1
```

GET Object Version ACL

The following request returns information, including the ACL, of a specific version of the object:

```
GET /{object-name}?version={etag}&acl HTTP/1.1
```

PUT Object ACL

PUT Object acl uses the acl subresource to send the ACL of an object in the request body (rather than in the request headers):

```
PUT /{object-name}?acl HTTP/1.1

<AccessControlPolicy>
  <Owner>
    <ID>ID</ID>
    <DisplayName>EmailAddress</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>ID</ID>
        <DisplayName>EmailAddress</DisplayName>
      </Grantee>
      <Permission>Permission</Permission>
    </Grant>
    ...
  </AccessControlList>
</AccessControlPolicy>
```

PUT Object Version ACL

The following request returns information, including the ACL, of a specific version of the object:

```
PUT /{object-name}?version={etag}&acl HTTP/1.1
```

Gateway CORS for Buckets

- Enabling CORS on a Bucket
- CORSRule Elements

Gateway supports Cross-Origin Resource Sharing (CORS) so that you can make it possible for a specific bucket to be accessed by a web page in a different domain. You configure a bucket to allow cross-origin resource access by using CORS configuration rules. These are two common scenarios for using CORS:

- Outbound Access - You are hosting a website in a bucket, but you want those site pages to use stylesheets, images, and scripts that are managed elsewhere. Because browsers block such requests from within scripts, you need to configure your bucket to explicitly enable cross-origin requests.
- Inbound Access - You are hosting a public resource from your bucket. Because browsers require a CORS check (known as a preflight check), you need to configure the bucket to allow any origin to make these requests.

> See the W3C specification for CORS: http://www.w3.org/TR/cors/

> Note
> Currently, you set CORS configuration using S3, not SCSP. However, a browser accessing the bucket will receive the same CORS information in the response from both S3 and SCSP.

Enabling CORS on a Bucket

To configure your bucket to allow cross-origin requests, you create a CORS configuration, an XML document with up to 100 rules that identify the origins that can access your bucket, the operations (HTTP methods) to support for each origin, and other operation-specific information. You add the XML document as the cors subresource to the bucket.

For example, this cors configuration on a bucket has three rules (the CORSRule elements), which do the following:

1. Allow cross-origin PUT, POST, and DELETE requests from the `https://www.example1.com` origin and allow all headers in a preflight OPTIONS request through the Access-Control-Request-Headers header. In response to any preflight OPTIONS request, Gateway will return any requested headers.
2. Allow the same cross-origin requests as the first rule but to another origin, `https://www.example2.com`.
3. Allow cross-origin GET requests from all origins. The '*' wildcard character refers to all origins.

```
<CORSConfiguration>
   <CORSRule>
      <AllowedOrigin>http://www.example1.com</AllowedOrigin>
      <AllowedMethod>PUT</AllowedMethod>
      <AllowedMethod>POST</AllowedMethod>
      <AllowedMethod>DELETE</AllowedMethod>
      <AllowedHeader>*</AllowedHeader>
   </CORSRule>
   <CORSRule>
      <AllowedOrigin>http://www.example2.com</AllowedOrigin>
      <AllowedMethod>PUT</AllowedMethod>
      <AllowedMethod>POST</AllowedMethod>
      <AllowedMethod>DELETE</AllowedMethod>
      <AllowedHeader>*</AllowedHeader>
   </CORSRule>
   <CORSRule>
      <AllowedOrigin>*</AllowedOrigin>
      <AllowedMethod>GET</AllowedMethod>
   </CORSRule>
</CORSConfiguration>
```

The CORS configuration allows optional configuration parameters, as shown in this CORS configuration that allows cross-origin PUT and POST requests from http://www.example.com:

```
<CORSConfiguration>
   <CORSRule>
      <AllowedOrigin>http://www.example.com</AllowedOrigin>
      <AllowedMethod>PUT</AllowedMethod>
      <AllowedMethod>POST</AllowedMethod>
      <AllowedMethod>DELETE</AllowedMethod>
      <AllowedHeader>*</AllowedHeader>
      <MaxAgeSeconds>3000</MaxAgeSeconds>
      <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
      <ExposeHeader>x-amz-request-id</ExposeHeader>
      <ExposeHeader>x-amz-id-2</ExposeHeader>
   </CORSRule>
</CORSConfiguration>
```

CORSRule Elements

Element	Description
AllowedMethod	Specifies which of the following values is allowed: GET, PUT, POST, DELETE, HEAD
AllowedOrigin	Specifies the origins that you want to allow cross-domain requests from, for example, `http://www.example.com/`. The origin string can contain at most one * wildcard character, such as `http://*.example.com`. You can optionally specify * as the origin to enable all the origins to send cross-origin requests. You can also specify `https` to enable only secure origins.
AllowedHeader	Specifies which headers are allowed in a preflight request through the Access-Control-Request-Headers header. Each header name in the Access-Control-Request-Headers header must match a corresponding entry in the rule. Gateway will send only the allowed headers in a response that were requested. Each AllowedHeader string in the rule can contain at most one * wildcard character. For example, `<AllowedHeader>x-amz-*</AllowedHeader>` enables all Amazon-specific headers.
ExposeHeader	Identifies a header in the response that you want customers to be able to access from their applications (for example, from a JavaScript XMLHttpRequest object).
MaxAgeSeconds	Specifies the time in seconds that your browser can cache the response for a preflight request as identified by the resource, the HTTP method, and the origin. By caching the response, the browser does not have to send preflight requests if the original request is to be repeated.

Token-Based Authentication

In addition to HTTP Basic authentication, Content Gateway allows for the use of an optional token-based authentication. Token-based authentication works by performing a one-time HTTP Basic authentication request within the Management API or to a special URI path in the Storage API in order to receive a token. This token is used on subsequent requests as proof of the user's credentials.

Tokens have the following characteristics:

- They are always owned by the user that creates them except for tokens created by token administrators.
- They expire at a fixed time after creation; default is 24 hours if not specified.
- They may contain an optional S3 secret access key for use with the S3 protocol.
- They may contain optional metadata matching the prefix pattern: `x-custom-meta-*`
- The owner can list and delete their active tokens.
- The token administrators can list and delete any user's active tokens.

Application developers may prefer to make use of the Management API in order to create tenant tokens for storage domains that belong to a tenant. Storage domain tokens are created with the special URI defined by the `tokenPath` IDSYS attribute.

The following is an example excerpt from a root IDSYS configuration file that defines the token settings. Both the `cookieName` and `tokenPath` parameters must be defined in order to enable token-based authentication.

```
{
"ldap" : { ...
 "cookieName": "token",
 "tokenPath": "/.TOKEN/",
 "tokenAdmin": "superuser@admindomain.example.com"
 }
}
```

Tokens are delivered using the standard HTTP cookie mechanism. The `cookieName` parameter is the cookie's name and the value is the token. The token value is guaranteed to be universally unique and impossible to guess. The `tokenPath` parameter defines the URI path within the storage domain with which a user requests a token and then performs listing and delete operations on their active tokens. The `tokenAdmin` is the user name of the token administrator who is able to create, list, and delete tokens on behalf of other users.

The token administrator should be a fully qualified user name in order to avoid ambiguity in a situation where a storage domain may inherit its IDSYS from the tenant or root scope.

> See "Qualification of User/Group Names" in the IDSYS Document Format.

Gateway stores all tokens within the administrative domain as objects that automatically expire using the object lifepoint feature. The expiration time of an authentication token can be specified when the token is created. If the time is not specified, a default expiration time will be assigned based on the `tokenTTLHours` parameter in the `[gateway]` section of the gateway.cfg file. If an expired token is presented to Gateway, the request will proceed as an anonymous user subject to all of the normal access control policies. Additionally, the `Set-Cookie` header of the response will instruct the HTTP client to delete its expired token cookie.

To create a new authentication token, POST a blank document to either:

- your storage domain and token path, or
- the Management API path `/_admin/manage/tenants/{tenantName}/tokens/`

using HTTP Basic authentication to authenticate the request. Requests to the `tokenPath` URI are processed independently from the storage protocol handling and these instructions work with both SCSP and S3 front-end protocols and to the Management API.

Note

In these examples, HTTP Basic authentication is demonstrated using "Auth: {user}:{password}" for clarity. Be sure to use the Authorization HTTP request header according the definition in RFC 2717.

Creating Tokens

Query Arguments for Tokens

The following HTTP request URI query arguments control the creation of a token:

No query args	Causes the default behavior as if `setcookie=true` was specified.
setcookie=true	Causes the HTTP response to contain a Cookie header that will cause a web browser to replace its current authentication token with the newly generated one.
setcookie=false	Causes the HTTP response to contain the header Gateway-Token instead of the standard Cookie h eader. Use this to have the browser continue using its current authentication token. Note The Gateway-Token header is the same for both SCSP and S3 tokens.

Request Headers for Tokens

The following HTTP request headers control the creation of a token:

X-Owner-Meta:{username}	Required	Used by the `tokenAdmin` user to create a token on behalf of another user. An error is returned if any user other than the token administrator attempts to set this header. By default, the owner of a token will be the user that creates it.
X-User-Token-Expires-Meta:{time-specification}	Optional	Sets the expiration time for the authentication token. If this header is not given, the default expiration time is set based on Gateway's `tokenTTLHours` c onfiguration setting, which defaults to 24 hours after token creation.
X-User-Secret-Key-Meta:{string}	Optional	Sets an S3 secret key that is used with the S3 Protocol Personality for signing S3 requests. When this header is present, the token may only be used to sign S3 storage requests. Tokens with this header may not be used to authenticate SCSP storage or Management API operations. Values of this string must follow Swarm metadata value rules for encoding, and 7-bit ASCII values are recommended.
X-Custom-Meta-{string}:{string}	Optional	Additional custom metadata that is saved with the token. This is for application-specific purposes and it is not interpreted by the Gateway during token creation or use.

X-Custom-Meta-Description:{string}	Optional	This metadata header will be displayed as the token description in the Content Portal.

Token expiration time specification:

POSIX time	{n}	Integer value that is the number of seconds elapsed since 00:00:00. Coordinated Universal Time (UTC), 1 January 1970, not counting leap seconds. Example: "1444419929"
Days offset	+{n}	Integer number of days (86,400 sec/day) from now. Example: "+365"
Year only	{YYYY}	Four-digit year; the expiration will be on January 1st at 00:00Z of that year. Example: "2015"
Specific day	{YYYY}{MM}{DD}	Year, month, and day; the expiration will be at 00:00Z on that day. Example: "2015-10-09"
ISO timespec	{YYYY}{MM}{DD} T{hh}:{mm}:{ss}.{nnn}Z	ISO time specification; all digits and fixed characters must be supplied; only UTC ("Z") time zone is allowed. Example: "2015-10-09T11:18:00.000Z"

Token Examples

Creating a domain token

```
POST http://{domain}/.TOKEN/
Auth: john:password

HTTP/1.1 201 Created
Gateway-Request-Id: 41B8FD0D739DF86C
Set-Cookie: token=d9f8378f71e79b77831f65d9e6891af6; path=/
Content-Length: 0
```

Creating a tenant token for S3

```
POST http://{domain}/_admin/manage/tenants/tenant256/tokens/
Auth: john:password
X-User-Token-Expires-Meta: +730
X-User-Secret-Key-Meta: 5ZdMSEubcFHJjnkyEzy722ZQHjd2xsTo
X-Custom-Meta-Description: Laptop Applications

HTTP/1.1 201 Created
Gateway-Request-Id: 7612F7FDB63B7C02
Set-Cookie: token=cc8ea2467d196b047497818f6271f00c; path=/
Content-Length: 0
```

Creating a tenant token for S3 with curl

```
$ USER="john"
$ SECRETKEY="1NnYIOXeHfuuW30eARH19iJQXNvvjMSF"
$ EXPIRES="+365"
$ curl -u $USER -X POST --data-binary "" \
 -H "X-User-Secret-Key-Meta: $SECRETKEY" \
 -H "X-User-Token-Expires-Meta: $EXPIRES" \
 "http://mydomain.example.com/_admin/manage/tenants/tenant255/tokens/"
Enter host password for user 'john':
{"token":"8c3955185d3ae8347caca1a14e4e2416", ... }
```

Managing Tokens

- Listing Authentication Tokens
- Removing an Authentication Token
- Clearing Tokens for Locked Accounts
- Token Examples

Listing Authentication Tokens

In order to list your active authentication tokens, perform a GET on the token path using an existing authentication token or using HTTP basic authentication to validate the request.

Listing domain tokens

```
GET http://{domain}/.TOKEN/?format=json
Cookie: token=d9f8378f71e79b77831f65d9e6891af6

HTTP/1.1 200 OK
Gateway-Request-Id: F48303758301E570
Castor-Object-Count: 3
Content-Type: application/json; charset=utf-8
Content-Length: 651
[
  {"x_token_domain_meta":"{domain}", "x_owner_meta":"john",
   "last_modified":"2012-06-22T05:39:44.854100Z",
   "lifepoint":"[Sat, 23 Jun 2012 05:39:44 GMT] reps=2,[] delete",
   "name":"7e742e12fb7e070b44266df1a1bf2efe"},
   ...
]
```

Listing tenant tokens

```
GET http://{domain}/_admin/manage/tenants/tenant256/tokens/
Authorization: Basic Z2NhcmxpbjpmdW5ueQ==
```

Removing an Authentication Token

In order to logout and remove an authentication token, perform a DELETE on the full token path and authenticate the request with a token or with HTTP basic authentication.

Deleting a domain token

```
DELETE http://{domain}/.TOKEN/53dfb96dc6d5b9cacd174e3649cba6d5
Cookie: token=22f57e203c10cf86d2dfd9564b1413f5
```

Deleting a tenant token

```
DELETE
http://{domain}/_admin/manage/tenants/tenant256/tokens/53dfb96dc6d5b9cacd174e3649cba6d5
Authorization: Basic Z2NhcmxpbjpmdW5ueQ==
```

If you delete a token and use the same token that you are deleting to authenticate the request, the Gateway will return a `Set-Cookie` header to clear your token. This is useful when implementing logout pages for web browsers.

Deleting a domain token with itself

```
DELETE http://{domain}/.TOKEN/53dfb96dc6d5b9cacd174e3649cba6d5
Cookie: token=53dfb96dc6d5b9cacd174e3649cba6d5

HTTP/1.1 200 OK
Gateway-Request-Id: 9855371AA8411781
Set-Cookie: token=; path=/
Content-Length: 0
```

Note

When using the token in the URI path, the operation must be authenticated using either the token within a `Cookie` header or by using a valid user and password in an `Authentication` header with the request. The audit log will reflect the name of user that owns the token if the cookie is used or the name of the authenticated user if HTTP basic authentication is used.

Clearing Tokens for Locked Accounts

Because identity management systems are poor at signalling that an account has been locked, Gateway allows unexpired tokens to continue to work for locked accounts. For a removed account, the token will stop working as soon as it expires from cache.

However, for an account that is expired (locked) but not removed, extra measures are needed to ensure that its tokens stop working:

PAM Authentication:

This method is for those using a PAM as a front-end for traditional Unix authentication.

1. Lock the user account by change the password: `passwd -l USERNAME`
2. Change the username: `zzzUSERNAME`

LDAP Authentication:

1. Standardize an attribute within one of the schemas that apply to the user record for which enabled user accounts will always have set to a known value.
2. Design a test for the value.

> Tip
>
> Although you could use a negative test to find disabled accounts, there is less risk of mistakes with the affirmative method (attribute is value).

For example, you could use the pwdPolicy schema with the pwdLockout attribute and use the userFilter to require the pwdLockout attribute to be true.

Token Examples

The token administrator defined in the root IDSYS configuration file is allowed to use the `x-owner-meta` argument in order to perform token listing for any user. Administrators wishing to disable a user account and log them out of the system could do so by locking their LDAP account and then removing any existing authentication tokens for that user.

The following examples show how the token administrator lists and deletes another user's tokens.

Token administrator `superuser@admindomain.example.com` listing the authentication tokens for user `john:`

Discovering tokens

```
GET http://{domain}/.TOKEN/?format=json&x-owner-meta=john
Auth: superuser@admindomain.example.com:superpassword

HTTP/1.1 200 OK
Gateway-Request-Id: 29172D0FDCAB19DE
Castor-Object-Count: 1
Content-Type: application/json; charset=utf-8
Content-Length: 221
[
  {"x_token_domain_meta":"{domain}",
   "x_owner_meta":"john",
   "last_modified":"2012-06-24T07:14:53.671600Z",
   "lifepoint":"[Mon, 25 Jun 2012 07:14:53 GMT] reps=2,[] delete",
   "name":"b71805b6c862860bfed892c653cbc4b5"}
]
```

Using the tokens discovered during the listing operation, the token administrator then issues deletes for each of the tokens in exactly the same way the user would delete their own tokens.

Deleting token

```
DELETE http://{domain}/.TOKEN/b71805b6c862860bfed892c653cbc4b5
Auth: superuser@admindomain.example.com:superpassword

HTTP/1.1 200 OK
Gateway-Request-Id: 4628361DE8318726
Content-Length: 0
```

Notice that the token administrator lists tokens the same way any user does and is able to specify an arbitrary user with the `x-owner-meta` query argument. The delete operation is the same pattern whether performed by the user or the token administrator.

Best practice
Use the token administrator's credentials when accessing or deleting tokens for other users so that the audit log reflects that the token administrator performed the operations.

Gateway Audit Logging

Gateway's audit log of user actions is designed for machine parsing so that it can be used for auditing, compliance monitoring, API request analysis, and SLA reporting.

> For configuration of the logging output, see Gateway Configuration.

This section focuses on the format of the audit logs to allow for integration and development of applications that use them.

- Audit Log Message Fields
- Audit Log Message Format

Audit Log Message Fields

These are the fields that appear in logging output. These are only definitions and not the format of any particular log message.

Field Name	Description
Auth Domain	Tenant or storage domain name used to authenticate user; tenant names prefixed with "+"
Auth User	User ID used to authenticate; empty if anonymous
Bucket	Name of bucket
DNS Domain	Origin DNS domain; value of Host header from the request

Domain	Swarm domain name to which operation refers to
Elapsed Time	Transaction time in milliseconds
HTTP Code	Request response code. Exceptions in request handling return a 500. All SCSP requests that have authorization errors output a 401.
Log Level	Logging level for the audit log entry
Message Type	Message category to simplify filtering
Object Name or UUID	Named of object, excluding bucket name, or UUID for unnamed streams
Operation	The operation. Examples: POST, HEAD, DELETE
Record Format Version	Audit log record format version. This will change if format of the output records is different from the previous release.
Request ID	A unique identifier for client request attached to all associated audit messages. This value matches the HTTP response header Gateway-Request-Id given to the client and is used in the server log.
Response Bytes	Number of bytes sent to Source IP in the HTTP response body
Source Bytes	Number of bytes received from Source IP in the message body
Source IP	IP address from which a request originated
Timestamp	High resolution timestamp up to millisecond

Audit Log Message Format

Following are the output formats for all event types. All log messages share a common set of prefix fields, which includes a message type. The suffix fields in a log message are variable based on the message type. This allows for automated parsing of log messages.

The fields in each log message are separated by spaces. If a field value is missing, the string (none) is substituted. Field values are subject to HTML URL encoding in order to make spaces, UTF-8, and other special characters safe for inclusion in the audit log entry.

- Alphanumeric characters "a" through "z", "A" through "Z" and "0" through "9" remain unchanged
- Characters ".", "-", "*", and "_" remain unchanged

- Space character is converted into a plus sign "+"
- All other characters are converted into %HH byte values using UTF-8 encoding

> Note
> The "/" character in an object's name will appear as "%2F" in the log, based on the previous rules.

Common Prefix Fields

All messages will be prefixed by the following fields in this order:

1. Timestamp
2. Log Level
3. Request ID
4. Record Format Version
5. Source IP
6. DNS Domain
7. Message Type
8. Operation
9. Auth User
10. Auth Domain
11. HTTP Code
12. Source Bytes
13. Response Bytes
14. Elapsed Time

Suffix Fields

This table defines the suffix fields that are included with each log message following the common prefix fields.

Event	Message Type	Operation	Suffix Fields
User requests token	Auth	GET	
User deletes token		DELETE	
List available domains	Admin	LIST_DOMAINS	
Domain creation	Domain	POST	Domain
Domain policy create/ update		POLICY_PUT	
Domain policy read		POLICY_GET	
Domain policy delete		POLICY_DELETE	
Domain copy		COPY	

Domain delete		DELETE	
Domain read		GET	
Domain info		HEAD	
List buckets in a domain		LIST_BUCKETS	
Bucket creation	Bucket	POST	Domain, Bucket
Bucket policy create/ update		POLICY_PUT	
Bucket policy read		POLICY_GET	
Bucket policy delete		POLICY_DELETE	
Bucket copy		COPY	
Bucket delete		DELETE	
Bucket read		GET	
Bucket info		HEAD	
List objects in a bucket		LIST_OBJECTS	
S3 list multiparts		LIST_MULTIPARTS	
Object creation	Scsp	POST	Domain, Bucket, Object name or UUID
Object update		PUT	
Object append		APPEND	
Object copy		COPY	
Object delete		DELETE	
Object read		GET	
Object info		HEAD	
S3 multipart initiate		MULTIPART_INITIATE	Domain, Bucket, Object name
S3 multipart put		MULTIPART_PUT	
S3 multipart copy		MULTIPART_COPY	

S3 multipart abort		MULTIPART_ABORT	
S3 multipart complete		MULTIPART_COMPLETE	
S3 list multipart		LIST_MULTIPART	

Example Log Messages

These are examples of a variety of audit log messages.

Successful login for user muser1 to the domain nom.dom.com

```
2012-05-13 19:28:29,671 INFO [9D9A577B66D2DD56] 2 172.20.1.1 172.20.1.2
  Auth POST muser1 nom.dom.com 201 0 0 0.48
```

Successful POST of a bucket named redbucket by user admin1

```
2012-05-13 19:28:25,070 INFO [7169E3D6DD5656B9] 2 172.20.1.1 172.20.1.2
 Bucket POST admin1 nom.dom.com 201 0 44 0.65 nom.dom.com redbucket
```

401 authentication challenge on a HEAD to an unauthenticated request

```
2012-05-13 19:28:36,632 INFO [85822E93CFBC6F12] 2 172.20.1.1 172.20.1.2
 Bucket HEAD (none) nom.dom.com 401 0 0 0.72 nom.dom.com redbucket
```

Writing an object named water.jpg to bucket bluebucket without being required to authenticate

```
2012-05-15 14:54:31,616 INFO [D2AC19A94ECA5A51] 2 172.20.1.1 172.20.1.2
  Scsp POST (none) open.dom.com 201 10 44 1.05 open.dom.com bluebucket
water.jpg
```

Reading an object named water.jpg to bucket bluebucket without being required to authenticate

```
2012-05-15 14:54:31,818 INFO [86B6E646C65DC83B] 2 172.20.1.1 172.20.1.2
  Scsp GET (none) open.dom.com 200 0 10 1.12 open.dom.com bluebucket
water.jpg
```

Listing a bucket without being required to authenticate

```
2012-05-15 14:54:45,236 INFO [C87A09C1FCCCE581] 2 172.20.1.1 172.20.1.2
  Bucket LIST_OBJECTS (none) open.dom.com 200 0 273 2.57 open.dom.com
bluebucket
```

Listing a domain as user admin1

```
2012-05-15 16:32:14,560 INFO [CAE97BE991DE877A] 2 172.20.1.1 172.20.1.2
 Domain LIST_BUCKETS admin1 nom.dom.com 200 0 180 2.38 nom.dom.com
```

Administrative override and replacement of domain's Policy by user superuser from root IDSYS

```
2012-10-16 10:37:29,719 INFO [D580617E135E35DF] 2 172.30.1.1 172.20.1.2
 Domain POLICY_PUT !superuser@ nom.dom.com 201 123 0 1.08 nom.dom.com
```

Specific Operations

When a GET operation is interrupted, such as if the socket closed unexpectedly prior to reading all data, the audit log may record an HTTP 200 response with response bytes equal to the size of the object. When interruption takes place, an HTTP 500 response is logged with response bytes equal to the actual number of bytes that were transmitted.

If multiple messages are logged as a result of one client operation, all messages will have the same Request ID so that they can be correlated with the client request. For example, the recursive delete operation will generate synthetic delete requests all with the same Request ID.

Application-Supplied Tag

Gateway's audit logging allows for the client application to supply a custom tag that can be used to correlate multiple audit log entries to one application-level transaction. The application specifies this tag in a Gateway-Audit-Id request header and it must be alpha-numeric and is truncated at 32 characters. When this optional tag is received, the Request

ID field of the audit log entry will contain the automatically-generated request identifier from the Gateway, a dash ("-"), and the application-supplied tag.

Example of a normal request identifier and one with the application supplied tag trans123

```
2012-12-10 09:30:45,360 INFO [1813AC1764D48125] ...
2012-12-10 09:30:45,360 INFO [2AF5F226122D9673-trans123] ...
```

When the application-supplied tag is used for multiple operations, even across multiple Gateway servers, the request identifiers remain unique with a common suffix.

Gateway Practical Applications

This section offers practical applications of the Content Gateway features and concepts.

- Restricting Domain Access

Restricting Domain Access

When Gateway is deployed by a managed service provider, cluster administrators inevitably need to cut off some or all access to the hosted domains within their cluster. This could be due to non-payment or if a customer uses too much storage and is required to clean-up space before writing new content.

All access to a domain can be controlled from the root Policy configuration file and from the domain's policy attribute. Updating the policy attribute is often desirable because, unlike an update to the root Policy file, it does not require a Gateway server restart. These examples will use the policy attribute of a domain for controlling access. Recall that the statements in an access Policy have an optional Sid field that can be used in whatever way an application wants. When injecting statements into an existing Policy, administrators can use the Sid field to keep track of the statements they added and to identify them for future removal.

- No Access
- Read-Only Access
- Read- and Delete-Only Access

No Access

In this example, a domain that had allowed access to the domain administrator (one of the end-users) now completely cuts off access to all end-users by adding the deny statements. The new statements use the Sid field to identify them for easy removal in the future. Notice that the statement denies authenticated users as well as anonymous users.

```
{
   "Statement": [
      {
         "Resource": "/*",
         "Action": [
            "*"
         ],
         "Principal": {
            "user": [
               "domainadmin"
            ]
         },
         "Effect": "Allow"
      },
      {
         "Resource": "/*",
         "Action": [
            "*"
         ],
         "Principal": {
            "user": [
               ""
            ],
            "anonymous": [
               ""
            ]
         },
         "Effect": "Deny",
         "Sid": "temp-cutoff-noaccess"
      }
   ]
}
```

Read-Only Access

In this example, a domain is changed to read-only mode in order to prevent writing, updating, or deleting content by the-end users. The new policy statement makes use of the Sid field to identify it for future removal. This example also makes use of NotAction to specify that the deny pertains to any action not listed thus allowing the ones that are listed.

```
{
   "Statement": [
      {
         "Resource": "/*",
         "Action": [
            "*"
         ],
         "Principal": {
            "user": [
               "domainadmin"
            ]
         },
         "Effect": "Allow"
      },
      {
         "Resource": "/*",
         "NotAction": [
            "GetObject",
            "GetBucket",
            "GetDomain",
            "ListBucket",
            "ListDomain",
            "GetDomainPolicy",
            "GetPolicy",
            "PutPolicy"
         ],
         "Principal": {
            "user": [
               ""
            ],
            "anonymous": [
               ""
            ]
         },
         "Effect": "Deny",
         "Sid": "temp-cutoff-ro"
      }
   ]
}
```

Read- and Delete-Only Access

A cluster administrator could set the access control policy on a domain for read and delete only if a tenant exceeds their quota. By letting the end-users continue to read and delete their content, they can use the content they have already written and clean-up content in order to reduce their storage usage. As with the previous example, NotAction is used to specify that the deny pertains to any action not listed.

```
{
   "Statement": [
      {
         "Resource": "/*",
         "Action": [
            "*"
         ],
         "Principal": {
            "user": [
               "domainadmin"
            ]
         },
         "Effect": "Allow"
      },
      {
         "Resource": "/*",
         "NotAction": [
            "GetObject",
            "GetBucket",
            "GetDomain",
            "ListBucket",
            "ListDomain",
            "GetDomainPolicy",
            "GetPolicy",
            "PutPolicy",
            "DeleteObject",
            "DeleteBucket",
            "DeleteDomain"
         ],
         "Principal": {
            "user": [
               ""
            ],
            "anonymous": [
               ""
            ]
         },
         "Effect": "Deny",
         "Sid": "temp-cutoff-readdelete"
      }
   ]
}
```

Migrating Applications from Swarm Storage

This section describes the changes that are required when adapting native Swarm storage applications to use Content Gateway.

Requirements

- Supply storage domain name in all requests
- Use HTTP basic authentication instead of digest
- Use Gateway ACL system instead of native Swarm auth/auth
- Do not use Integrity Seal hash-type upgrade through Gateway

> Tip
>
> When integrating with Gateway, applications do not need to handle the HTTP 100-continue or redirect semantics that Swarm clients must include: the Gateway operates as a reverse proxy and will correctly use 100-continue when communicating with Swarm and hides all redirects from the upstream client.

Domains

Because Gateway is performing access control and validation for all operations, every content request must identify the storage domain for which the request is destined. The order of precedence for specifying the storage domain is:

1. Query argument: domain=X, else
2. HTTP X-Forwarded-Host header, else
3. HTTP request Host header value.

While some native integrations with Swarm are rigorous in specifying the storage domain, Swarm is permissive of requests that do not specify one. Swarm also has additional precedence rules for assigning the storage domain; these are not compatible with requests handled through Gateway. When using Gateway, an application must specify the storage domain explicitly using one of the listed methods.

Authentication

Because Gateway is often deployed in access-controlled environments, it is common to require client applications to authenticate their requests. While applications that previously integrated with Swarm may not have chosen to include provisions for authenticating their requests, it is required to provide for HTTP basic authentication when integrating with the Content Gateway.

> Deprecated
>
> The native Swarm auth/auth feature is deprecated and will be removed after June 2017. If you are using Swarm's native auth/auth for your applications, you must add `security.noauth = False` now in order to continue using the native auth/auth.

Applications can interoperate with Gateway and Swarm by implementing the Gateway ACL system or using a library that provides for an automatic selection. Unless an application manipulates the access control policies within Swarm, no additional changes are required when integrating with Gateway. Applications that do manipulate these policies will need to be adapted for Gateway's enhanced access control mechanism.

SSL

Content Gateway provides system administrators with the capability of encrypting client communications with SSL. Applications should provide for HTTPS communications when integrating with Gateway. Since many HTTP libraries already provide this capability, it is likely that applications will only need to add a configuration provision to use HTTPS versus HTTP.

S3 Protocol Interface

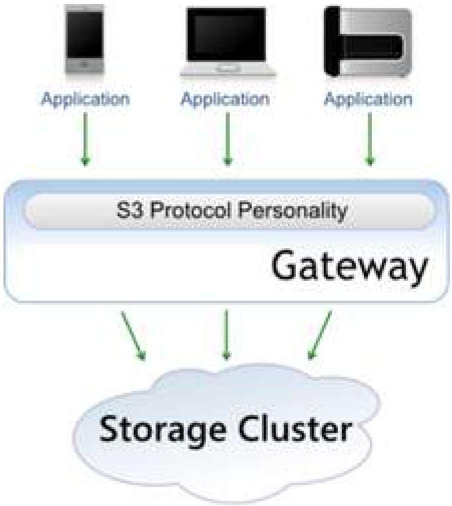
This section covers the software configuration of the S3 object storage protocol and provides guidance for integrating existing AWS S3 applications. Information in this document builds upon Content Gateway Implementation and Content Application Development.

- S3 Protocol Architecture
- S3 Protocol Configuration
- S3 Application Integration
- Supported Amazon S3 Features
- S3 Protocol Special Topics

S3 Protocol Architecture

- Sharing Storage across S3 and SCSP
- Routing Methods

The S3 protocol personality is a front-end storage protocol for client applications. It runs within the Gateway itself. All of the Gateway deployment and scaling features apply when you use the S3 front-end protocol.



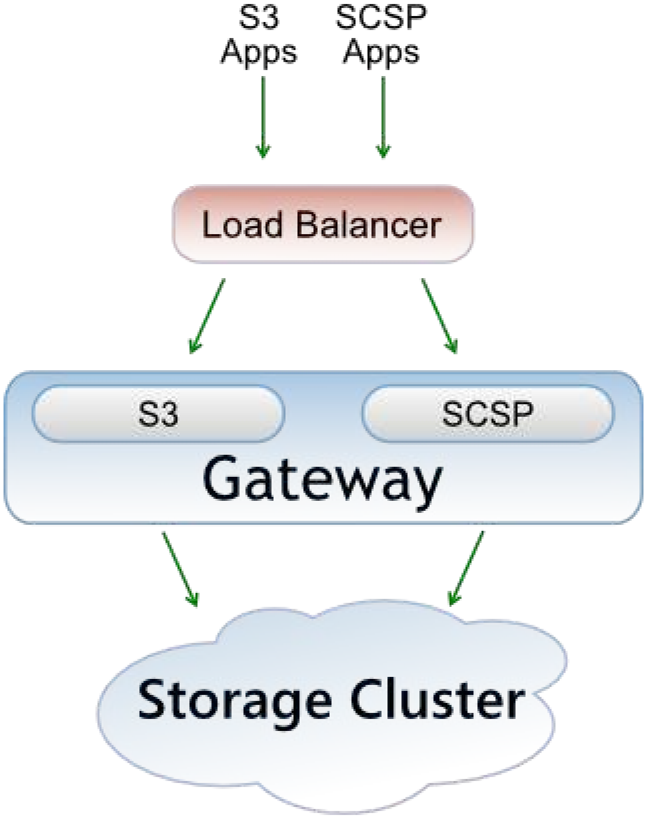
An administrator can configure the Gateway to run several ways:

- only the SCSP protocol
- only the S3 protocol
- both protocols at the same time

Additionally, Gateways can be scaled horizontally with any combination of front-end protocols as needed for a particular deployment. This allows for a heterogeneous mix of client types that utilize the same Swarm storage cluster and that share content with each other. Doing this allows an administrator to provide a unified object storage platform.

Sharing Storage across S3 and SCSP

Content Gateway provides the mechanism to unify the back-end Swarm object storage so that S3 applications and SCSP applications can share content. In a unified object storage deployment, a device like a load balancer is used to route incoming client traffic through the appropriate port number or pool of Gateway servers.



Routing Methods

These are some routing methods that can be used for a unified object storage front-end:

IP address	Listen to multiple virtual IP addresses and distribute traffic based upon the IP address used by the client.	

X-Forwarded-Host	Use Layer 7 inspection of the requests to distribute traffic based upon headers contained in the client requests.	The X-Forwarded-Host header can be used by a proxy or load balancer to let client applications use a different host name for each supported HTTP storage protocol while sharing the same storage domain for their content. For example, consider the storage domain `castor.example.com` with two protocol-specific host names `scsp.castor.example.com` and `s3.castor.example.com`. Each host would resolve to a different IP address so that a load balancer could direct the traffic to the appropriate Gateway server pool for that chosen storage protocol. In order to direct the storage requests to the shared storage domain, the load balancer would need to add the header: `X-Forwarded-Host: castor.example.com`
DNS	Cause the DNS name resolution to be different for the clients using one storage protocol than for the clients using another protocol.	For example, the clients using S3 may resolve the storage domain castor.example.com to 10.100.100.81 while clients using SCSP would resolve the same storage domain to 10.100.100.82. While this method does not require in-line modification of the HTTP requests, it does require that the administrator have control of the hosts where the client applications run so as to allow alteration of the DNS/host resolution.

OPTIONS	Route request method OPTIONS with "Origin" to the S3 port rather than default to the SCSP port (which happens when an Authorization header does not exist or have "AWS").	S3 must handle "bucket in Host" style requests because SCSP would report that the domain was not found.
		`curl -i`
		` -H 'Origin: http://www.example.com'`
		` -H 'Access-Control-Request-Method: PUT'`
		` -X OPTIONS`
		`https://mycorsbucket.elsewhere.com/`

Pattern-matching	Use Layer 7 inspection of the requests to switch incoming traffic based on the object storage protocol. See below:	This is done by looking for the distinctive S3 Authorization header pattern or one of the query string arguments for authenticated S3 requests. The pattern-matching rules for these authenticated S3 requests are as follows.

• Headers	Use pattern-matching on the request header.	`Authorization ~= ^AWS.*`
• Arguments	Use pattern-matching on the query string arguments.	`{RequestURL} ~= [?&](AWSAccessKeyId|X-Amz-Credential)=`
• Absence	Use pattern-matching to test for the absence of all AWS request patterns.	The request is either an anonymous S3 request or an SCSP request. Since anonymous S3 requests should not create, update, or delete content, they are most likely GET requests, and it is safe to allow SCSP to handle these.

S3 Protocol Configuration

In order to use the S3 front-end protocol, first configure the Gateway as described in Gateway Configuration and then perform these additional steps:

1. Verify that your Swarm storage configuration settings are correct, which is required for S3 clients to perform actions such as bucket deletion.
2. Edit the `gateway.cfg` file for S3 use:
   a. In the `[s3]` section, enable the S3 front-end protocol.
   b. In the `[storage_cluster]` section, define `indexerHosts` for at least one indexer server.
3. Create one or more authentication tokens for each S3 client.

When the S3 front-end protocol is in use, the Gateway must be able to query the Swarm Elasticsearch metadata index servers directly. If you have multiple metadata index servers, you can include as many as you wish in the `indexerHosts` parameter in order to spread the load across them and to provide fail-over in case one becomes unavailable.

The S3 protocol makes use of a shared secret key that is known to the client and the Gateway in order to provide request validation. The client creates an HMAC signature for every authenticated request and the Gateway must independently recreate the signature in order to validate the request. The AWS S3 access key and secret key is implemented with Gateway's token-based authentication.

S3 Application Integration

Configuring existing Amazon S3 applications to work with Swarm consists of changing the region end-point and changing the authentication credentials.

> Best practice
> Start with the documentation provided by Amazon Web Services and then use this section to help you integrate your S3 applications with the Swarm platform.

Within your S3 applications, change the following items:

1. Region end-point – Use the Swarm storage domain name in place of the Amazon S3 region end-point host name.
2. Access Key – Create an S3 authentication token and use the token ID as your Access Key ID.
3. Secret Key – From the same S3 authentication token and use its secret key value in your S3 applications. For creating tokens in Content Portal, see Setting Tokens.

Supported Amazon S3 Features

This table summarizes the Amazon S3 features that are supported by the Gateway's S3 protocol implementation.

> Note
> When you are listing uploads and there are multiple simultaneous uploads in progress for a single object, only one of the uploads will be in the listing.

Scope	Supported Operational Feature
Error Responses	HTTP Response Errors
Common Request Headers	Authorization (AWS Signature Versions 2 and 4) Content-Length Content-MD5 Content-Type Date Expect Host x-amz-date
Common Response Headers	Connection Content-Length Content-Type Date ETag Server x-amz-delete-marker x-amz-request-id
Service	GET Service

Buckets	DELETE Bucket DELETE Bucket cors DELETE Bucket policy GET Bucket GET Bucket acl GET Bucket cors GET Bucket Location GET Bucket Object versions GET Bucket policy GET Bucket versioning HEAD Bucket List Multipart Uploads PUT Bucket PUT Bucket acl PUT Bucket cors PUT Bucket policy PUT Bucket versioning Cross-Region Replication (via Swarm replication feed)
Objects	DELETE Object DELETE Multiple Objects GET Object GET Object acl HEAD Object PUT Object PUT Object acl PUT Object - Copy Initiate Multipart Upload Upload Part Upload Part - Copy Complete Multipart Upload Abort Multipart Upload List Parts Query String Request Authentication (pre-signed URLs)

S3 Protocol Special Topics

Amazon S3 is two distinct things: "S3 The Service" and "S3 The Protocol." Since the S3 protocol reflects characteristics of the Amazon service that may not be applicable outside of Amazon, the Gateway adapts these characteristics so that they make sense when hosting storage within your environment. These adaptations are transparent to most applications and enhance the protocol features by making use of the unique strengths of Swarm storage.

- Regions and Storage Domains

Regions and Storage Domains

Amazon S3 currently has eleven geographic regions where buckets are stored and each region is shared by thousands of end-users. A customer selects a region for their content based upon latency, cost, and any applicable regulatory requirements. Although there are multiple Amazon S3 regions, customers must choose bucket names that are unique across all regions. Although AWS S3 now allows for bucket replication between regions, the name of the source bucket and destination bucket must be different due to the globally unique bucket names rule.

Within the S3 Protocol Personality, the concept of an Amazon S3 region is mapped to storage domains in Swarm. Because any number of storage domains can be created in Swarm, this allows for more granular assignment to users and it allows for automatic geographic distribution of domains in multiple locations. When using remote replication in Swarm, content within a domain can be created, updated, and accessed in multiple storage clusters with automatic synchronization being handled by the storage cluster. This includes the ability for local and geographic distribution. The domain names, bucket names, and object names will be identical to every cluster to which they are replicated. This allows for content sharing and direct DR fail-over.

> Best practice
>
> Domain creation and allocation in Swarm is lightweight, so give each customer or business unit their own storage domain. Doing so makes storage tracking and management by the storage administrators easier, and it gives end-users simpler access control policies and greater flexibility when naming their buckets (which need to be unique within their domain only).

Bucket Location

The Amazon S3 GET Bucket Location request returns the AWS region in which the bucket is located. This request in the Gateway's S3 protocol implementation returns the value of the `cluster.name` parameter configured in Swarm. If a storage domain exists in more than one cluster, the return value for a bucket location request will depend upon the cluster that serves the request.

Unlike Amazon S3 where the geographic location of a bucket is chosen when it is created and stays fixed, Swarm cluster placement for a storage domain and the buckets it contains is controlled by the storage administrator. Additionally, a domain may exist in multiple storage clusters if the administrator has setup remote replication.

> Best practice
>
> Give every cluster a unique cluster name so that applications can use the name to identify the location where their content is being served.

Storage Class

Amazon S3 allows clients to set a storage class preference in the `x-amz-storage-class` header, which defines the data durability and access frequency of content:

- STANDARD
- REDUCED_REDUNDANCY
- STANDARD_IA
- GLACIER

The S3 protocol tags all objects with the `x-amz-storage-class-meta` header and includes the client application's requested class or STANDARD, if none is specified. Bi-directional translation between the AWS S3 `x-amz-storage-class` header and the Swarm `x-amz-storage-class-meta` header is done for S3 protocol operations.

Virtual Hosting of Buckets

Amazon S3 allows virtual host name to bucket mappings within their storage service. This is accomplished by creating a DNS alias (CNAME record) for a virtual host name that to points one of the Amazon S3 region end-points. An example would be to allow the web request to be mapped to the real Amazon S3 URL:

```
http://www.fred.com/hello.html
http://s3.amazonaws.com/www.fred.com/hello.html
```

If the virtual host is mapped to the bucket named `www.fred.com` in the US Standard Region, then the HTTP `Host` header can be used to specify the bucket name.

The S3 protocol also supports this mapping of virtual hosts to buckets. To accomplish this, the storage administrator configures their DNS server to perform wildcard resolution of host names to the front-end IP address of the Gateway. The Gateway will then search for a storage domain within Swarm starting with the value of the `Host` header and then recursively popping off the leftmost word using period (".") as a delimiter until it finds a match or runs out of words. When a storage domain is found, the previously popped words are concatenated back together using periods and the result becomes the bucket name for the request.

As an example, consider the storage domain in Swarm called `fred.com` that contains a bucket called `www` and an object within that bucket called `hello.html`. The normal method to access this object is with the URL `http://fred.com/www/hello.html`, which has the following HTTP request headers:

```
GET /www/hello.html HTTP/1.1
Host: fred.com
```

In order to setup the virtual host mapping from `www.fred.com` to the Swarm storage domain and bucket, a DNS entry must be setup for `www.fred.com` and `fred.com`. This is an example where the Content Gateway's front-end IP address is `10.100.100.81` and a wildcard match is used.

```
fred.com    A      10.100.100.81
*.fred.com  CNAME  fred.com
```

After the DNS entries are in place, the `hello.html` object is now accessible with the additional URL `http://www.fred.com/hello.html`. The HTTP request headers that arrives at the Gateway look like this:

```
GET /hello.html HTTP/1.1
Host: www.fred.com
```

The Gateway first checks for the non-existent storage domain `www.fred.com` and then removes `www`, the leftmost word, and finds the storage domain `fred.com` in the storage cluster. The Gateway then transparently modifies the HTTP request headers by prefixing the URI path with the removed word `www` and shortening the `Host` header as follows:

```
GET /www/hello.html HTTP/1.1
Host: fred.com
```

If the bucket name contains periods, `hires.images` with a virtual host name of `hires.images.fred.com` , the Gateway will search for a storage domain by removing the leftmost words until a domain is found or until it reaches a null host name. For example, it would test for the existence of the domains `hires.images.fred.com` and `images.fred.com` before finding `fred.com`. The request results in an error if the search reaches a null host name. The `[caching]domainExistenceRefresh` configuration parameter in `gateway.cfg` is used to optimize domain existence testing.

Identity Management and S3 Authentication Tokens

The Gateway's S3 protocol makes use of an external identity management system (IDM) for users and groups, similar to federation with AWS IAM, and uses an internal system for managing authentication tokens, similar to AWS temporary security credentials. These authentication tokens are created and managed within the Swarm cluster.

Authenticated requests to the S3 protocol follow the AWS S3 request signing rules for v2 and v4 signatures whereby each request includes a header in one of the following forms:

- `Authorization: AWS AccessKey:Signature` (v2 signature)
- `Authorization: AWS4-HMAC-SHA256 Credential,SignedHeaders,Signature` (v4 signature)

The construction of the Authorization header is automatically handled by S3 SDKs and S3 applications. For the complete details on the elements of the header's value string and the S3 HMAC authentication mechanism, consult the online AWS S3 documentation.

In order to authenticate S3 protocol requests to the Gateway, each S3 client needs at least one authentication token. The Access Key value is the ID of an authentication token.

> See Token-Based Authentication.

When creating authentication tokens for use with S3 (also referred to as "S3 authentication tokens"), the `X-User-Secret-Key-Meta` header is required when creating the token object. The value of this header becomes the Secret Access Key (or just "Secret Key") that is used to sign S3 requests. As previously mentioned, the Access Key becomes the token cookie's value returned by the create request.

This example shows an authentication token being created by the user `gcarlin` with an S3 secret key of `abcdefg` and an expiration time of 365 days from now. Note that this request uses HTTP basic authentication to create the token.

```
POST /.TOKEN/ HTTP/1.1
Authorization: Basic Z2NhcmxpbjpmdW5ueQ==
Host: abc.cloud.example.com
X-User-Secret-Key-Meta: abcdefg
X-User-Token-Expires-Meta: +365
Content-Length: 0
```

This is an excerpt from the Gateway's response.

```
HTTP/1.1 201 Created
Set-Cookie: token=722bfb49aa8365897a3e774d539038ce; expires=Fri,
06-Jun-2017 18:44:52 GMT; path=/
```

In order to sign S3 requests with this newly created token, construct the S3 Authorization header using:

```
AccessKey=722bfb49aa8365897a3e774d539038ce
SecretAccessKey=abcdefg
```

> Note
>
> Tokens that contain an S3 secret key may only be used to sign S3 storage requests and may not be used to authenticate SCSP storage operations.

Error Response for Missing Resource

In order to provide additional details for troubleshooting errors regarding non-existent content, the S3 protocol includes an extra XML `Resource` tag in the error response. This is an example showing the additional field.

```
<Error>
 <Code>NoSuchKey</Code>
 <Message>The specified key does not exist.</Message>
 <Resource>/mybucket/missingFile.txt</Resource>
 <RequestId>03B8CD915CD6C3A5</RequestId>
 <Key>missingFile.txt</Key>
</Error>
```

The format of the `Resource` tag is: "/{bucketName}/{objectName}".

Multipart Uploads

AWS S3 requires that every part (except for the last one) of a multipart upload must be at least 5MB in size. The Gateway's S3 protocol implementation does not impose this limitation and allows each part of a multipart upload to be of any size.

List after Update Timing

After an object is created, the delay before that object appears in a list operation can vary depending upon the Swarm metadata feed batch timeout setting. When the batch timeout is set to 1 second, new objects are typically available within two seconds following a create. The Amazon S3 documentation has specific developer guidance about this eventual consistency behavior.

PUT Object Copy Metadata

The AWS S3 PUT Object Copy request makes a duplicate of an existing object and, when the `x-amz-copy-source` header is included with the request, will copy the `x-amz-meta-*` custom metadata from the source object.

Although Swarm allows for more custom metadata patterns than this, only the metadata matching `x-amz-meta-*` pattern will be copied during a PUT Object Copy operation.

S3 Versioning

Swarm's native object versioning feature is interoperable with AWS S3 versioning. The implementation includes these improvements:

- Ability to disable versioning:
  AWS S3 only allows for versioning to be suspended once enabled on a bucket. Swarm provides the ability to disable versioning and automatically clean up the prior versions in order to reclaim storage space.
- Delete marker consolidation:
  Unlike AWS S3 where continued DELETE operations on a deleted object will record additional delete markers in the version history, Swarm will acknowledge the subsequent deletes without recording additional delete markers. Multi-factor authentication delete is not supported.
- Expanded version listing:
  Swarm supports version listing batches up to 2000 items while AWS S3 limits these listing results to batches of

1000. Additionally, Swarm does not break batches on version boundaries. Delimiter case is currently not supported for version listing.

- Simplified ACL management:
  When using per-object ACLs with versioning, the ACL for the current version of the object applies for determining authorization. To change the ACL for an object's entire version chain, update the object without specifying a version.

Swarm SDK

- SDK Overview
- SDK for Cpp
- SDK for Csharp
- SDK for Java
- SDK for Python

SDK Overview

- Getting Started
  - Installation
  - Run-Time Configuration
  - Logging
  - Errors and Status Codes
  - Common Terminology
  - Using SDK Code Examples
- ScspClient
  - SCSP Classes
  - Support Classes
  - Validation Mode
- Managing Domains and Buckets
- Object Headers and Query Arguments
  - Using ScspHeaders for System and Custom Metadata
  - ScspQueryArgs

The Swarm Software Development Kit (SDK) simplifies integration with Swarm by providing client library support for handling of specific SCSP behaviors to programmers developing Swarm applications. A client application communicates with Swarm using a subset of the HTTP/1.1 protocol called Simple Content Storage Protocol (SCSP). SCSP implements all of the required components and most of the common methods of the HTTP protocol and includes the Swarm specific handling of standard conventions like URL query arguments and custom request headers.

The SDK describes a consistent set of features using a common API in each of the following programming languages:

- C++
- C#
- Java
- Python

All example clients are synchronous and thread-safe. High performance applications can call any of the clients from multiple threads and/or multiple processes without interference or deadlocks.

Note
You must review the Storage SCSP Development prior to using the Software Development Kit for a full understanding of the SCSP protocol and recommendations for client integration. This guide assumes prior knowledge of both Swarm and the basic requirements for a native client that communicates with Swarm.

Getting Started

Installation

You can install and run the SDK on any operating system that supports the specific programming language that you will be using (Java, Python, C#, or C++).
The Software Development Kit (SDK) distribution ZIP file contains the following:

- Source code for each language implementation (see the language-specific sections for source tree location/contents)
- Language-specific documentation (such as javadocs)
- Language-specific code examples for all SCSP methods and several commonly used query arguments and headers
- Third-party utilities that are required to compile or run a SDK SCSP client or which provide useful utilities
- LICENSE.txt - Swarm SDK SCSP Client license

Run-Time Configuration

Configuring SDK Timeouts

The SDK enables you to configure the following timeouts:

- ConnectionTimeout: For Java, Python and C#, sets the length of time an open request socket can be inactive. For C++, the amount of time the client will wait for a connection to be opened. C++ also has an additional requestTimeout for the time limit, in secs/GB, for the request to complete. For instance, with a requestTimeout of 200 secs/GB, a 2GB write request times out after 400 seconds.
- PoolTimeout: Sets the amount of time an open connection stays in the connection pool before being closed.
- LocatorRetryTimeout: Sets the amount of time between attempts in the locator to retry a node discovered to be unavailable.

Using Connection Pooling

The connection pool stores open, previously used connections for reuse so your client does not have to negotiate opening a socket for every request. The following table shows which parameters control connection pooling for each supported programming language. Caringo recommends you set the value of each parameter shown in the table to the number of threads multiplied by the number of Swarm cluster nodes.

Language	Parameter name
C++	maxStoredConnections
C#	maxStoredConnections

Java	maxConnectionPoolSize
Python	maxSavedConnections

> Important
> For installations with a large number of nodes and a high thread count (approximately 100 or more), be sure to limit the value to 5 times the number of threads to avoid reaching the client's operating system limits on open file descriptors.

Using Static Location and Locator Types

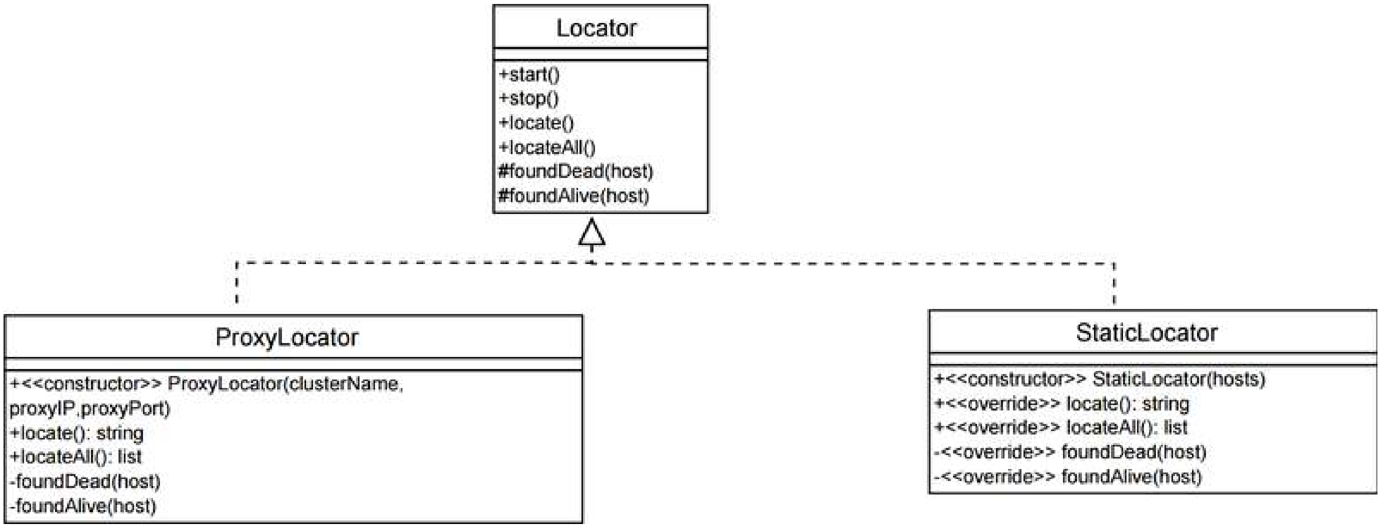
There is a new ProxyLocator subclass available in each SDK programming language that performs the following functions:

1. Performs a GET / to the SCSP Proxy to pre populate its local list of Swarm node IP addresses.
2. Dynamically maintains this list as redirects and other responses are received directly from Swarm nodes.

The ProxyLocator class API is basically identical to the existing StaticLocator, with the exception that the constructor accepts the following additional parameters:

- SCSP Proxy IP address or host name
- SCSP Proxy's port
- Swarm cluster name

The following UML diagram shows the API of the Locator interface and its subclasses, ProxyLocator and StaticLocator:


Locator
+start()
+stop()
+locate()
+locateAll()
#foundDead(host)
#foundAlive(host)
ProxyLocator
+<<constructor>> ProxyLocator(clusterName,
proxyIP,proxyPort)
+locate(): string
+locateAll(): list
-foundDead(host)
-foundAlive(host)
StaticLocator
+<<constructor>> StaticLocator(hosts)
+<<override>> locate(): string
+<<override>> locateAll(): list
-<<override>> foundDead(host)
-<<override>> foundAlive(host)

If the SDK node list is currently empty, the locate() method throws an exception that includes reason field of the Scsp-Proxy-Nodes response header. For more information about Scsp-Proxy-Nodes, see the SCSP Proxy Overview Guide.

Using the HostHeaderValue Property

To access a named object, you can supply a HostHeaderValue property on ScspClient that overrides the Host header in the request. HostHeaderValue specifies the object's domain name.

If an ScspClient method includes an ScspHeader argument that includes a Host header, ScspClient sends it to Swarm using that Host header. If the Host header is empty but the HostHeaderValue property on ScspClient is not empty, ScspClient uses that value for the Host header.

If both are empty, ScspClient uses the host in the request URL.

Using the Path Argument

The Path argument has changed in SDK version 1.2 to support named objects, unnamed objects, and SCSP Proxy paths.

The following syntax is valid only with the SCSP Proxy; sending requests formatted as follows directly to a Swarm cluster results in a 404 (Not Found) error because Swarm attempts to resolve it as a named object.

/_proxy is a required prefix for accessing objects using the SCSP Proxy.

> Note
> The uuid parameter is deprecated and replaced by path in this release. The uuid parameter will be removed in a future release.

Syntax	Description
/_proxy/*cluster-name*/*uuid-or-name*	Sends a request for an object, referenced by UUID or by name, to a specific cluster name.
/_proxy/all/*uuid-or-name*	Queries all configured clusters (remote and local) for a particular object, determines the current version, and returns that object to the client.
/_proxy/any/*uuid-or-name*	**any** is valid only for remote INFO and results in an error if used with any other method. **any** causes a request to be sent for an object, referenced by UUID or by name, to any available cluster (local or remote). If the object exists in the local cluster the information is returned. Otherwise, the request is sent to each remote cluster in random order. If no cluster is able to locate the data, the error response from the local server is returned.
/_proxy/remote/*uuid-or-name*	**remote** is valid only for remote INFO and results in an error if used with any other method. **remote** causes a request for an object to be sent, referenced by UUID or by name, to a remote cluster. The information is returned from the first cluster that has the object. If the object cannot be found in any of the remote clusters, the error response that was received from the first remote cluster tried will be returned.

> Deprecated
> Support for remote SCSP Proxy requests without the /_proxy prefix is deprecated and will be removed altogether in a future release.

Logging

The SDK does not currently provide a standard logging mechanism; however, some implementation libraries, such as the HttpClient package on which the java SDK is built, have built-in logging that can provide useful information. Where applicable, the language implementations provide examples of how to turn on the built-in logging for advanced users.

Errors and Status Codes

The API returns error and status codes or throw exceptions as appropriate to the implementation language. Where possible, when the API throws an exception it throws SCSP-specific exceptions, although there are certain exception states that are hard to fully anticipate in any language or library.

Common Terminology

Throughout this document and in other documents describing Swarm characteristics, the following terms are used interchangeably:

- Object (also referred to as a stream or a file) is a piece of unstructured content stored in a Swarm cluster.
- Client (also referred to as an application) is a system or a particular instance of a system that accesses a remote service on a server, Swarm in this instance, by way of a network.

Using SDK Code Examples

All four languages supported by the SDK now offer samples that create, update, and delete objects; and execute other SCSP methods on them.

Before you run these samples, you must verify all of the following:

Tenants

Your cluster administrator must create two tenants as follows:

Domain name	Domain protection setting
allusers.realm	All users. No authentication required
localusers.realm	Only users in this domain

In addition, neither domain can have any domain managers.

Contact your cluster administrator to perform these tasks and make sure the administrator sets up the tenants exactly as shown in the preceding table.

Credentials

Your client application must specify a valid user name and password for a user in the CAStor administrator realm.

By default, Swarm has a CAStor administrator user named admin with the password `ourpwdofchoicehere`. No action is necessary to use these defaults.

If you changed the administrator password, you must edit the example file before compiling and running it.

Contact your cluster administrator to get this information.

Local Environment

Caringo recommends you run these samples in a non-production environment.

To set up the proper local environment, you can either edit the Environment.* file for the language you are using or you can set the following local environment variables:

Variable	Meaning
SCSP_HOST	Host name or IP address of a node in the Swarm cluster. This host or IP address must be accessible from the machine on which you run the examples.
SCSP_PROXY_HOST	Used by the remote examples only. Host name or IP address of the local SCSP Proxy.
SCSP_PORT	Swarm cluster node's SCSP listen port. Default is 80.

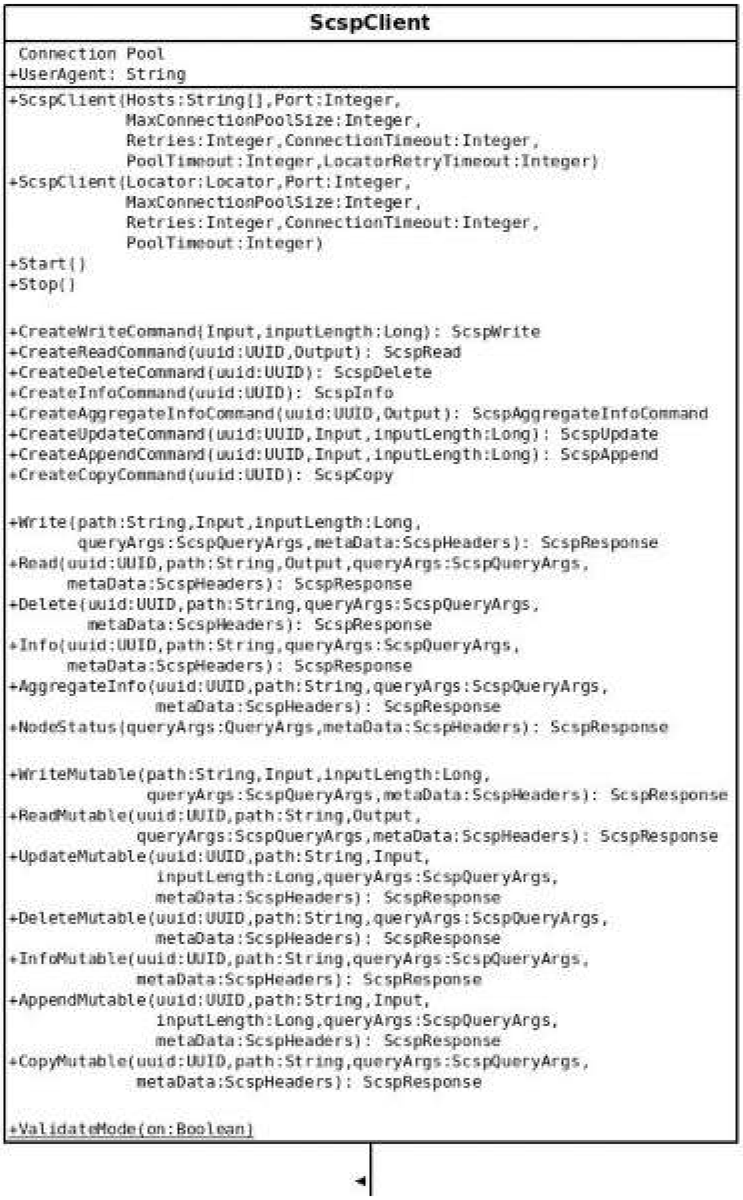
ScspClient

ScspClient is a collection of execution classes that provides procedural methods for execution of the various SCSP commands. The API supports the following commands:

- Unnamed objects
    - Write, Read, Info, Delete (both mutable and immutable)
    - Update, Copy, Append (mutable only)
- Named objects (which are always mutable): Write, Read, Info, Delete, Copy, Update, and Append
- AggregateInfo
- Node Status

In all, Swarm supports the following general types of objects:

- Immutable unnamed objects, which can be deleted but not changed. If you delete an unnamed object, its UUID is not reused.
- Mutable unnamed objects (that is, anchor streams), which have contents that can be replaced but UUIDs that never change. Anchor streams must have the ?alias=yes query argument. Like immutable unnamed objects, an anchor stream's UUID is not reused after the object is deleted.
- Named objects, which are mutable but which are addressed by name instead of by UUID. If you delete a named object, another object with the same name can be created later.



```
ScspClient

 Connection Pool
+UserAgent: String

+ScspClient(Hosts:String[],Port:Integer,
            MaxConnectionPoolSize:Integer,
            Retries:Integer,ConnectionTimeout:Integer,
            PoolTimeout:Integer,LocatorRetryTimeout:Integer)
+ScspClient(Locator:Locator,Port:Integer,
            MaxConnectionPoolSize:Integer,
            Retries:Integer,ConnectionTimeout:Integer,
            PoolTimeout:Integer)
+Start()
+Stop()

+CreateWriteCommand(Input,inputLength:Long): ScspWrite
+CreateReadCommand(uuid:UUID,Output): ScspRead
+CreateDeleteCommand(uuid:UUID): ScspDelete
+CreateInfoCommand(uuid:UUID): ScspInfo
+CreateAggregateInfoCommand(uuid:UUID,Output): ScspAggregateInfoCommand
+CreateUpdateCommand(uuid:UUID,Input,inputLength:Long): ScspUpdate
+CreateAppendCommand(uuid:UUID,Input,inputLength:Long): ScspAppend
+CreateCopyCommand(uuid:UUID): ScspCopy

+Write(path:String,Input,inputLength:Long,
       queryArgs:ScspQueryArgs,metaData:ScspHeaders): ScspResponse
+Read(uuid:UUID,path:String,Output,queryArgs:ScspQueryArgs,
      metaData:ScspHeaders): ScspResponse
+Delete(uuid:UUID,path:String,queryArgs:ScspQueryArgs,
        metaData:ScspHeaders): ScspResponse
+Info(uuid:UUID,path:String,queryArgs:ScspQueryArgs,
      metaData:ScspHeaders): ScspResponse
+AggregateInfo(uuid:UUID,path:String,queryArgs:ScspQueryArgs,
               metaData:ScspHeaders): ScspResponse
+NodeStatus(queryArgs:QueryArgs,metaData:ScspHeaders): ScspResponse

+WriteMutable(path:String,Input,inputLength:Long,
              queryArgs:ScspQueryArgs,metaData:ScspHeaders): ScspResponse
+ReadMutable(uuid:UUID,path:String,Output,
             queryArgs:ScspQueryArgs,metaData:ScspHeaders): ScspResponse
+UpdateMutable(uuid:UUID,path:String,Input,
               inputLength:Long,queryArgs:ScspQueryArgs,
               metaData:ScspHeaders): ScspResponse
+DeleteMutable(uuid:UUID,path:String,queryArgs:ScspQueryArgs,
               metaData:ScspHeaders): ScspResponse
+InfoMutable(uuid:UUID,path:String,queryArgs:ScspQueryArgs,
             metaData:ScspHeaders): ScspResponse
+AppendMutable(uuid:UUID,path:String,Input,
               inputLength:Long,queryArgs:ScspQueryArgs,
               metaData:ScspHeaders): ScspResponse
+CopyMutable(uuid:UUID,path:String,queryArgs:ScspQueryArgs,
             metaData:ScspHeaders): ScspResponse

+ValidateMode(on:Boolean)
```

To support the new large objects feature in Swarm version 6.0, methods in the ScspClient class now support the value UNDETERMINED_LENGTH for an object's inputStreamLength. UNDETERMINED_LENGTH can be used with an object (such as a live video feed) whose size is not known and causes an object to be sent to Swarm using standard HTTP chunked transfer coding.

In addition, all SDK languages support query arguments for erasure-coded objects: ?checkIntegrity, ?encoding, and ?segmentSize.

> For more information about erasure coded objects, see Erasure Coding EC.

SCSP Classes

The following sections briefly discuss the SCSP operations supported by the SDK in all languages. See the language-specific sections for additional operations that may be supported.

Write	The Write method writes an object to the specified cluster and returns a name or UUID. It is equivalent to an HTTP POST. A successful Write method execution returns an HTTP response code of either 201 or 202.
Read	The Read method returns an object requested by its name or UUID from a specified cluster. It is equivalent to an HTTP GET. A successful Read method execution returns an HTTP response code of 200. A successful Read with range headers might return a 206 response code. A successful Read with cache coherency (for example, if-match) headers might return a 304 Not Modified response code.
Delete	The Delete method deletes (if policy allows) an object identified by its name or UUID from a specified cluster. It is equivalent to an HTTP DELETE. A successful Delete method execution returns an HTTP response code of 200.
Info	The Info method returns the metadata for an object identified by its name or UUID from a specified cluster. It is equivalent to an HTTP HEAD. A successful Info method execution returns an HTTP response code of 200.
AggregateInfo	The AggregateInfo method is an extension of the normal SCSP operations. It must be used in requests to the SCSP Proxy; it cannot be used in requests directly to Swarm nodes. In a single request, you can get a list of names or UUIDs to be Infoed to the specified cluster using the following format: {uuid [mutable|immutable] | url-encoded-name} You must supply a list of either URL-encoded names or UUIDs. (Use percent encoding for object names, if needed.) For unnamed anchor streams, you must use the mutable parameter. (The default is immutable.) The name, UUID, or consistency checkpoint is stored as a Swarm object. Then in the AggregateInfo method, Info requests are issued for each name or UUID in the consistency checkpoint and either object metadata or an error response is returned in the concatenated response body. Similar to the individual Info method, the response for a successful AggregateInfo method execution is a 200 code. The AggregateInfo method supports named and unnamed objects for both the manifests and streams stored in the manifest. It supports authentication only for the manifest stream itself, and checkpoint streams in the manifests that are protected for HEAD are returned as 401s in the AggregateInfo response body.
NodeStatus	The NodeStatus method returns basic capacity information for the cluster as well as some high level SCSP operations counts for the queried node in the response body. The total and available capacity numbers are also returned as headers on the response, Castor-System-TotalGBCapacity and Castor-System-TotalGBAvailable respectively. In addition to the status data, this method can be useful in verifying an SDK client is talking to a live Swarm node. A successful NodeInfo method execution returns an HTTP response code of 200.

WriteMutable	The WriteMutable method writes a mutable object to the specified cluster and returns a name or UUID. It is equivalent to an HTTP POST with an alias=yes query argument for anchor streams. A successful WriteMutable method execution returns an HTTP response code of either 201 or 202. Note Anchor streams are not intended to be used as general-purpose, updateable objects. In particular, rapid reads or updates (more than once per second) of an anchor stream can produce unpredictable results and might even result in errors being logged or error responses being returned to the application.
ReadMutable	The ReadMutable method returns a mutable object requested by its name or UUID from the specified cluster. It is equivalent to an HTTP GET with an alias=yes query argument for anchor streams. A successful ReadMutable method execution returns an HTTP response code of 200 or 206.
UpdateMutable	The UpdateMutable method updates an existing mutable object in the specified cluster. It is equivalent to an HTTP PUT. A successful UpdateMutable method execution returns an HTTP response code of either 201 or 202.
DeleteMutable	The DeleteMutable method deletes (if policy allows) a mutable object identified by its name or UUID from a specified cluster. It is equivalent to an HTTP DELETE with an alias=yes query argument for unnamed anchor streams. A successful DeleteMutable method execution returns an HTTP response code of 200.
InfoMutable	The InfoMutable method returns the metadata for a mutable object identified by its name or UUID in the specified cluster. It is equivalent to an HTTP HEAD with an alias=yes query argument for unnamed anchor streams. A successful InfoMutable method execution returns an HTTP response code of 200.
AppendMutable	The AppendMutable method allows appending of new data on to the end of the content for an existing mutable object in the specified cluster. There is no HTTP equivalent for Append. A successful AppendMutable method execution returns an HTTP response code of either 201 or 202.
CopyMutable	The CopyMutable method allows metadata update without modifying the content of an existing mutable object in the specified cluster. There is no HTTP equivalent for Copy. A successful CopyMutable method execution returns an HTTP response code of 201 or 202.

Support Classes

Support classes are also available to implement common functions for ScspClient. The following support classes provide building blocks for creation of some of the components needed for various commands:

ScspIntegritySeal	This class provides integrity seal hash types for setting up write headers as well as integrity seal response parsing.

ScspAuthentication	This class provides a way to return a realm in response headers and to send a user name and password with request headers.
ScspAuthorization	Enables you to use authentication with Swarm 5.0 and later
ScspDate	This class provides the basic functionality for converting to and from language-specific dates to SCSP date formats, for both query arguments and Lifepoints.
ScspLifepoint	This class provides the components for building a single Lifepoint.
ScspDeleteConstraint	This class enumerates the standard Lifepoint delete constraint strings: deletable=no, deletable=yes, and delete.

> See the language-specific sections (SDK for Csharp, SDK for Cpp, SDK for Java, SDK for Python) for specifics on how each language implements the support classes.

Validation Mode

ScspClient can be run in validation mode to ensure proper formatting of some of the most commonly used query arguments and headers. The SDK can be run in Validation mode without being connected to any SCSP server or SCSP Proxy, in which case the following is validated.

Validation	Read	Info	Write	Delete	Update	Copy	Append
Header: Allow			V		V	V	
Header: Content- Type			V		V	V	
Header: Content- Disposition			V		V	V	
Header: Host	V	V	V	V	V	V	V
Query argument: replicate			V		V	V	V
Query argument: alias	V	V	V	V	V	V	V
Query argument: validate	V						
Query argument: hashtype	V		V		V	V	V
Query argument: hash	V						
Query argument: newhashtype	V						
Query argument: countreps		V					
Query argument: domain	V	V	V	V	V	V	V
Query argument: admin	V	V	V	V	V	V	V

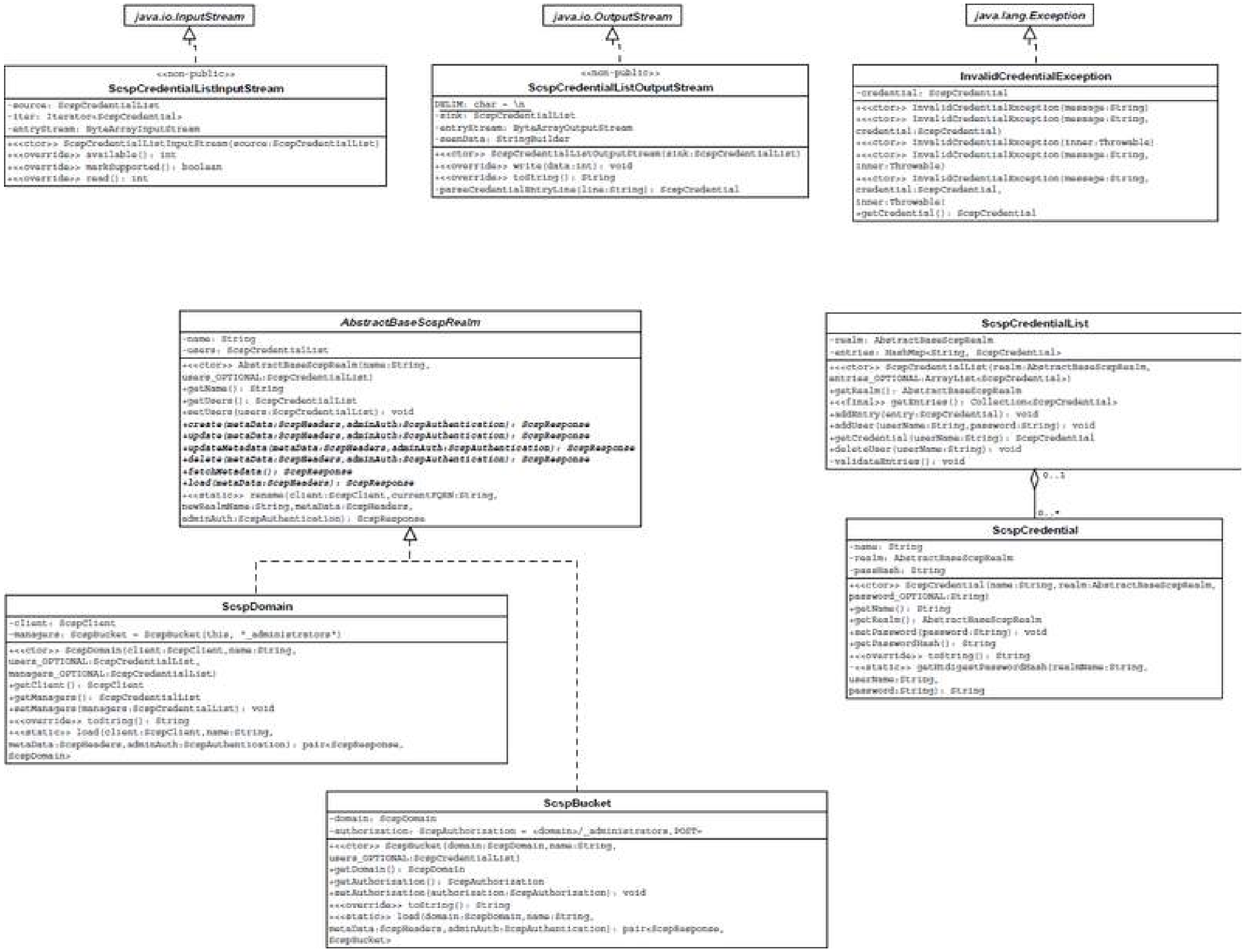
Managing Domains and Buckets

To assist you in managing domains (also referred to as tenants) and buckets, the SDK enables you to add, rename, or delete domains. For more information about named objects and tenancy, see the Swarm Application Guide.

The source code referred to in this appendix is located as follows:

- C++: sdk-extract-dir /cpp/src/realm
- C#: sdk-extract-dir \csharp\ScspCSExamples\ScspRealmExamples.cs
- Java: CAStorSDK-src-extract-dir /com/caringo/realm
- Python: castorsdk-python-egg-extract-dir /castorsdk/realm

Refer to the following UML diagram:


java.io.InputStream
java.io.OutputStream
java.lang.Exception
«non-public»
ScspCredentialListInputStream
-source: ScspCredentialList
-iter: Iterator<ScspCredential>
-entryStream: ByteArrayInputStream
+«ctor» ScspCredentialListInputStream(source:ScspCredentialList)
+«override» available(): int
+«override» markSupported(): boolean
+«override» read(): int
«non-public»
ScspCredentialListOutputStream
DELIM: char = \n
-sink: ScspCredentialList
-entryStream: ByteArrayOutputStream
-scanData: StringBuilder
+«ctor» ScspCredentialListOutputStream(sink:ScspCredentialList)
+«override» write(data:int): void
+«override» toString(): String
-parseCredentialEntryLine(line:String): ScspCredential
InvalidCredentialException
-credential: ScspCredential
+«ctor» InvalidCredentialException(message:String)
+«ctor» InvalidCredentialException(message:String,
credential:ScspCredential)
+«ctor» InvalidCredentialException(inner:Throwable)
+«ctor» InvalidCredentialException(message:String,
inner:Throwable)
+«ctor» InvalidCredentialException(message:String,
credential:ScspCredential,
inner:Throwable)
+getCredential(): ScspCredential
AbstractBaseScspRealm
-name: String
-users: ScspCredentialList
+«ctor» AbstractBaseScspRealm(name:String,
users_OPTIONAL:ScspCredentialList)
+getName(): String
+getUsers(): ScspCredentialList
+setUsers(users:ScspCredentialList): void
+create(metaData:ScspHeaders,adminAuth:ScspAuthentication): ScspResponse
+update(metaData:ScspHeaders,adminAuth:ScspAuthentication): ScspResponse
+updateMetadata(metaData:ScspHeaders,adminAuth:ScspAuthentication): ScspResponse
+delete(metaData:ScspHeaders,adminAuth:ScspAuthentication): ScspResponse
+fetchMetadata(): ScspResponse
+load(metaData:ScspHeaders): ScspResponse
+«static» rename(client:ScspClient,currentFQRN:String,
newRealmName:String,metaData:ScspHeaders,
adminAuth:ScspAuthentication): ScspResponse
ScspCredentialList
-realm: AbstractBaseScspRealm
-entries: HashMap<String, ScspCredential>
+«ctor» ScspCredentialList(realm:AbstractBaseScspRealm,
entries_OPTIONAL:ArrayList<ScspCredential>)
+getRealm(): AbstractBaseScspRealm
+«final» getEntries(): Collection<ScspCredential>
+addEntry(entry:ScspCredential): void
+addUser(userName:String,password:String): void
+getCredential(userName:String): ScspCredential
+deleteUser(userName:String): void
-validateEntries(): void
0..1
0..*
ScspCredential
-name: String
-realm: AbstractBaseScspRealm
-passHash: String
+«ctor» ScspCredential(name:String,realm:AbstractBaseScspRealm,
password_OPTIONAL:String)
+getName(): String
+getRealm(): AbstractBaseScspRealm
+setPassword(password:String): void
+getPasswordHash(): String
+«override» toString(): String
-«static» getMdigestPasswordHash(realmName:String,
userName:String,
password:String): String
ScspDomain
-client: ScspClient
-managers: ScspBucket = ScspBucket(this, "_administrators")
+«ctor» ScspDomain(client:ScspClient,name:String,
users_OPTIONAL:ScspCredentialList,
managers_OPTIONAL:ScspCredentialList)
+getClient(): ScspClient
+getManagers(): ScspCredentialList
+setManagers(managers:ScspCredentialList): void
+«override» toString(): String
+«static» load(client:ScspClient,name:String,
metaData:ScspHeaders,adminAuth:ScspAuthentication): pair<ScspResponse,
ScspDomain>
ScspBucket
-domain: ScspDomain
-authorization: ScspAuthorization = <domain>/_administrators,POST>
+«ctor» ScspBucket(domain:ScspDomain,name:String,
users_OPTIONAL:ScspCredentialList)
+getDomain(): ScspDomain
+getAuthorization(): ScspAuthorization
+setAuthorization(authorization:ScspAuthorization): void
+«override» toString(): String
+«static» load(domain:ScspDomain,name:String,
metaData:ScspHeaders,adminAuth:ScspAuthentication): pair<ScspResponse,
ScspBucket>

The SDK provides the following classes:

- ScspDomain, which creates, modifies, and deletes the domain object.

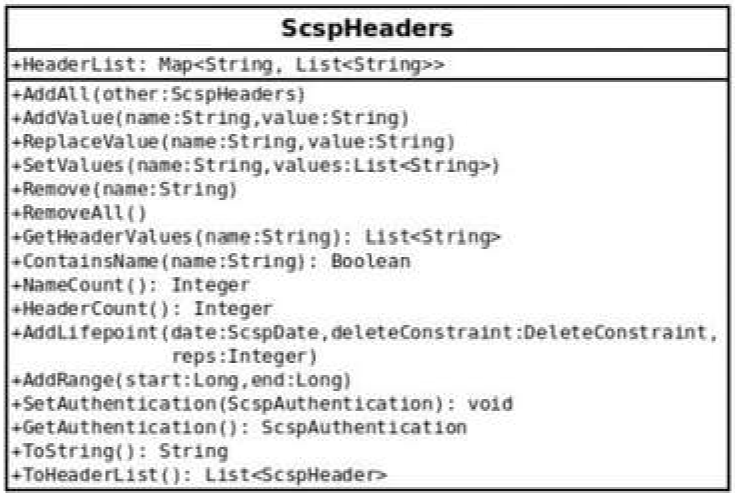
Note

> To avoid having the same domain created by more than one administrator, you should execute the methods in ScspDomain using a single thread.

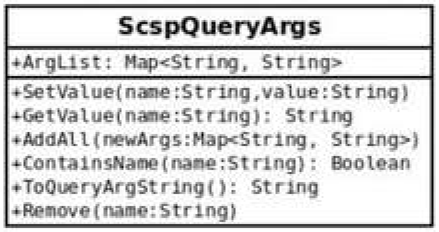
- ScspBucket, which creates buckets. ScspBucket has an attribute called (get/set)authorization that will control the Castor- Authorization header for the bucket.
- ScspCredential, which holds the authorization specification for a specific user.
- ScspCredentialList, which holds the list of credentials that are defined for a given user list.
- ScspCredentialListInputStream, which enables you to read the credential list using an InputStream interface.
- ScspCredentialListOutputStream, which enables you to build the credential list using an OutputStream interface.
- InvalidCredentialException, which is thrown whenever an invalid credential has been detected (such as a user list mismatch).

Object Headers and Query Arguments

The basic methods supported by the API can be extended with the addition of both standard HTTP request and Swarm-specific headers and/or query arguments. These optional components are separated from the methods themselves to easily support both custom headers as well as the addition of new headers and query arguments to the SCSP protocol over time.



```
ScspHeaders
+HeaderList: Map<String, List<String>>
+AddAll(other:ScspHeaders)
+AddValue(name:String,value:String)
+ReplaceValue(name:String,value:String)
+SetValues(name:String,values:List<String>)
+Remove(name:String)
+RemoveAll()
+GetHeaderValues(name:String): List<String>
+ContainsName(name:String): Boolean
+NameCount(): Integer
+HeaderCount(): Integer
+AddLifepoint(date:ScspDate,deleteConstraint:DeleteConstraint,
             reps:Integer)
+AddRange(start:Long,end:Long)
+SetAuthentication(ScspAuthentication): void
+GetAuthentication(): ScspAuthentication
+ToString(): String
+ToHeaderList(): List<ScspHeader>
```



```
ScspQueryArgs
+ArgList: Map<String, String>
+SetValue(name:String,value:String)
+GetValue(name:String): String
+AddAll(newArgs:Map<String, String>)
+ContainsName(name:String): Boolean
+ToQueryArgString(): String
+Remove(name:String)
```

Using ScspHeaders for System and Custom Metadata

About SCSP Headers

In HTTP and SCSP, metadata for requests, responses, and content itself are all represented by line-oriented, textual headers that prefix any binary data included with the message. The ScspHeaders class eases the creation and parsing of common header value syntax, enabling you to associate any string values with any header name you choose.

All SCSP client language implementations included in this SDK support standard HTTP request headers like Content-Type, Content-Length, Content-MD5, and Content-Disposition. The included clients also enable applications to use Swarm-specific headers, including Lifepoints and Castor-Authorization.

Any number of headers can be created but they are not filtered or validated in any way by default. When run in validation mode, the SDK validates Content-Type, Content- Disposition, and Allow headers. Multi-value headers are supported and can be created using either of the standard HTTP mechanisms for defining headers:

- A single header name with multiple string values
- Multiple entries for the same header name with different string values

All language implementations of the SDK, except C#, return headers verbatim from the Swarm response. In C#, because of the underlying HTTP header handling in .NET, the SDK splits header values into multiple header entries (except for Lifepoints, any headers with a name that includes 'date', and the Castor-System-Created header) into multiple header entries.

About the CAStor-system-* Header

Castor-System-* headers are reserved for internal use and are therefore not allowed on an incoming client request. If such a header is present in a request, the header is silently ignored.

ScspQueryArgs

HTTP query arguments are key/value pairs that are passed in the request along with the URL. Similar to ScspHeaders, the ScspQueryArgs object allows arbitrary association of names and string arguments as well as the following Swarm-specific arguments.

Argument	Description	Commands
`alias=yes`	Mutable unnamed objects (that is, anchor streams)	Write, Read, Info, Copy, Update, Append
`domain=`*`domain-name`*	Named objects only	Write, Read, Info, Copy, Update, Append, Delete
`replicate=immediate`	Replicate on write	Write, Copy, Update, Append
`validate=yes`	Check stored entity digest while reading	Read
`hashtype=`*`hash-algorithm`*`,hash=`*`digest`*	Compute and return an integrity seal	Write, Read, Copy, Update, Append
`newhashtype=`*`hashalgorithm`*	Compute and return an integrity seal	Read
`countreps=yes`	Return the current number of replicas for this name or UUID	Info

In addition, ScspQueryArgs allows callers to pass in arbitrary query arguments that are merged with any automatically generated ones and passed along with the request URL.

> See SCSP Query Arguments for all of the query arguments possible for each SCSP command.

SDK for Cpp

- C++ SDK Installation and Packaging
- Building the C++ Client on Linux
- Building the C++ Client on Windows
- Implementation Notes

- Using the C++ Client Sample Code
- C++ Administrative Override of an Allow Header

C++ SDK Installation and Packaging

The C++ distribution contains the following directory structure:

- `docs`: Contains HTML documentation for the C++ implementation
- `examples`: Contains example source and makefile which should compile and run against the supplied libraries and headers.
- `src`: Contains source and makefile
  - requestHandler: single-request HTTP communication engine
- `ScspCPPWin`: Contains the Visual Studio project for the Windows C++ implementation

Building the C++ Client on Linux

The C++ client requires these libraries and utilities:

- libcurl
- GNU Compiler Connection (gcc)

libcurl Requirements for the C++ Client

The C++ client requires a particular version of libcurl for your operating system.

The Swarm SDK is built on libcurl, version 7.20.1 or later. Version 7.21.2 was used for primary testing. libcurl is available pre-packaged or in source from http://curl.haxx.se/libcurl/. Get any of the version 7.20.1 or later source archives listed at the top of the curl download page.

Not all platforms ship with a recent version of libcurl by default. Specifically:

- SUSE: The packaged curl for SUSE 11.2 is version 7.20.1.6, which is supported by the SDK; however the additional libcurl-devel package must be installed.
- Windows 2003 and 2008: Get a recent version here.

GNU Compiler Connection (gcc) Requirement for the C++ Client

- gcc version 4 or later is required for Linux. Version 4.4.0 was used in primary testing.

Compiling the C++ Client on Linux

To build the source for a Linux target platform, you can use shell scripts that call make files.

To build the source for Red Hat Enterprise Linux (RHEL) or CentOS, run the following from the src directory:

- ./makeAllRH from the src directory

To build the source for SUSE, run the following from the src directory :

- ./makeAllSU

Building the C++ Client on Windows

Caringo tested the SDK with Microsoft Visual Studio Express 2008. If you use a different version of Visual Studio, additional tasks might be required. Consult the documentation provided with Visual Studio for more information.

Compiling curl on Visual Studio Express 2008

To compile curl on Visual Studio Express 2008:

1. Download curl version 7.20.1.7 or later. curl 7.20.1.7 is available here, and the most recent version of curl is available on the curl download page.
2. Open the curl Visual Studio project (for example, it might be named `vc6curl.dsw`).
3. You must convert the project to Visual Studio 2008 format.
4. Build curl.
   Because the procedure to compile curl changes frequently, consult an online reference such as the libcurl install page for more information.
5. After the solution builds successfully, close the project.

Compiling the Swarm SDK Using Visual Studio Express 2008

To compile the SDK using Visual Studio Express 2008:

1. Extract the SDK .zip file into an empty folder.
2. Open the Visual Studio solution (for example, `ScspCPPWin.sln`).
3. In the Solution Explorer pane, right-click the ScspCPPWin solution.
4. From the pop-up menu, click Properties.
5. From the Configuration list, click Debug.
6. Expand Configuration Properties > C/C++ > General.
7. In the Additional/Include Directories field, browse to locate the libcurl include directory.
8. Apply the changes and build the solution.
9. Repeat these tasks for your release project.

Compiling Your Custom Application Using Visual Studio Express 2008

This section provides basic guidelines for compiling your custom application with Visual Studio Express 2008. Caringo does not recommend any particular coding style or method; however, you must include the following tasks in your workflow for the compilation to succeed.

To compile your custom application using Visual Studio Express 2008, perform the following tasks in your debug and release projects:

1. Right-click the solution in the Solution Explorer pane.
2. From the pop-up menu, click Properties.
3. Expand Configuration Properties > C/C++ > General > Additional Includes.
4. Include the following:
   - Path to the curl include folder.
   - Path to the ScspCPP\src folder.
5. Expand Linker > General > Additional Library Directories.
6. Include the following:
   - Path to the curl lib folder.
   - Path to the ScspCPP\ScspCPPWin\Debug or ScspCPP\ScspCPPWin\Release folder.
7. Expand Linker > Input > Additional Dependencies.
8. Include paths to all of the following:

- ws2_32.lib wldap32.lib ScspCPPWin.lib
- libcurld.lib (debug) or libcurl.lib (release)

Implementation Notes

Using Connection Pooling with the C++ Client

The connection pool stores open, previously used connections for reuse so your client does not have to negotiate opening a socket for every request. You configure connection pooling for the C++ client for using the maxStoredConnections parameter in the ScspClient class. Caringo recommends you set the value of this parameter to the number of threads multiplied by the number of Swarm cluster nodes.

For more information about this parameter, see the scsp::ScspClient Class Reference topic in the HTML documentation provided with the SDK, which is located in the cpp/docs/ html directory in the SDK package.

> Note
>
> For installations with a large number of nodes and a high thread count (approximately 100 or more), you should limit the value to 5 times the number of threads to avoid reaching the client's operating system limits on open file descriptors.

Notes About the C++ SDK

Following are things to keep in mind when writing C++ code for the SDK:

- ScspClient chunkSize parameters support in C++. ScspClient in all languages, including C++, supports the following parameters: getChunkSize, setChunkSize. However, unlike other languages, curl does not support explicitly setting how many bytes are sent at a time. curl provides a buffer and the buffer's length but does not enable the SDK to set the size of the buffer.
- Compilation Warnings for scspCredential.ccp. When compiling C++ on Visual Studio, unsafe warnings related to conversion of the stream size argument for scspCredentials may print. These errors can be safely ignored.
- Expect headers. Because of a limitation of libcurl, to include more than one Expect header in a header entry, include them as a comma-separated list such as the following:

  ```
  wHeaders.addValue("Expect", "Content-MD5,100-Continue");
  ```

  Do not send each Expect header as a separate request because a large number of SCSP WARNING: Please use Expect: 100-continue... errors can result.
- Character encoding.
  - If you pass in a URI path, you must escape a backslash character (\) with %5c.
  - When you pass a path as a string, you must use the string class. If the path needs includes non-ASCII characters, these characters must be UTF-8 encoded by the caller.
- Synchronization Classes. To keep the SDK independent of a particular threading and synchronization package, the synchronization primitives used internally have been surfaced. SDK clients must derive lock and semaphore objects from the ScspLock and ScspSemaphore and their respective factories and pass instances of these into the ScspClient constructor. There is an example null lock and semaphore implemented in examples/ ScspExamples.cpp. Also, there is a ScspBasicLock in src/locator.hpp that implements a ScspLock derivative

based on a user-implemented ScspSemaphore derivative for ease of programming. See the documentation for locator.hpp or the Synchronization module in HTML help for more information.

- Pointer Ownership. Header and HTML documentation notes cover ownership of a pointer to an object. For the SDK, ownership of a pointer implies memory management responsibility. If an object owns a pointer to another object, it is responsible for deleting the object. For example, the pointers to the ScspLockFactory and ScspSemaphoreFactory passed into the constructor of an ScspClient are owned by the client and are deleted when the ScspClient instance is deleted.
- ScspResponse. Most of the ScspClient commands take a pointer to a ScspResponse as an argument for performance optimization. This response object is used internally by the SDK to set query result information. See the ScspClient documentation for more information.
- Exceptions. By design, C++ does not throw exceptions to indicate error conditions. Any exceptions thrown are due to system signals converted to exceptions by the gcc runtime.
- Mimetype Discovery. curl does not support mimetype discovery, so the C++ client also does not.
- Scsp Streaming. The SDK defines two interfaces, scsp_istream and scsp_ostream, that encapsulate all the required behavior from an input stream, from which content is retrieved for storage into a Swarm stream, or an output stream, into which to retrieve the content of a Swarm stream. Implementations are provided that wrap a std::iostream and std::ostream. scsp_istream and scsp_ostream are simpler to implement than a custom Standard C++ <iostream> streambuf and they can support large (>2GB) streams, even on 32-bit systems.
- Libcurl Versions. Versions of libcurl earlier than 7.20.1 do not fully support all components necessary to the SDK and are therefore not certified for use.
- Request Timeouts. In addition to the connectionTimeout, the C++ implementation uses a requestTimeout for the time limit, in secs/GB, for the request to complete. For instance, with a requestTimeout of 200 secs/GB, a 2GB write request will timeout after 400 seconds.

Using the C++ Client Sample Code

The C++ SDK client ships with runnable sample code that demonstrates how to write client applications for Swarm. The C++ sample code is located in the `sdk-zip-extract-dir/caringo-sdk-version-brand/cpp/examples` directory and consists of the following files:

C++ sample code file name	Description
ObjectEnumerator-Example.cpp	ObjectEnumerator class examples for use with the Content Router. Content Router has been deprecated and these examples will be removed in a future release.
ScspExample.cpp	General SCSP client examples.
ScspRemoteExample.cpp	Examples of performing SCSP operations on local and remote clusters using the SCSP Proxy.

Exploring the C++ Sample Code

The `sdk-extract-dir/SwarmSDK-version/docs/example` directory contains commented C++ sample code you can view in a web browser. Double-click index-all.html to open the index page in your default web browser. The commented samples include the following:

- Authenticating named objects
- Performing SCSP operations on unnamed mutable and immutable objects
- Using lifepoints and MD5 integrity seals
- Using READ, including validation, ranges, and Content-MD5

Performing SCSP operations on objects in local and remote clusters using the SCSP Proxy
You can also download the commented sample code from Caringo Connect.

C++ Administrative Override of an Allow Header

This example is not included with the code samples provided with the SDK.

```
// This example shows how to execute a copy command with administrative
// override to clean up Allow headers. Note that you may want to get the
// original headers first to make sure none of the existing ones on the
stream
// are lost.

string DEFAULTADMINREALM = "CAStor administrator";
isw.seekg(0, ios::beg);
ScspHeaders headers;
// set the admin credentials
ScspAuthentication auth("admin", "ourpwdofchoicehere",
DEFAULTADMINREALM);
headers.setAuthentication(auth);
ScspQueryArgs args;
// make the request administrative
args.setValue("admin", "yes");
client.copy(uuid, &response, &args, &headers);
```

SDK for Csharp

- C# SDK Installation and Packaging
- Required Environment
- Implementation Notes
- Using the C# Client Sample Code
- C# Administrative Override of an Allow Header

C# SDK Installation and Packaging

The C# distribution contains the following directory structure:

- `ScspCS`
    - `Properties`: the properties of the Visual Studio project file
    - Visual Studio project (`ScspCS.csproj`)

  - source files: `*.cs` files for each of the major components
- `ScspCSExamples`: code examples for all major functions

Required Environment

The C# client requires the following libraries and utilities:

1. Microsoft Visual Studio C# 2013 or Visual Studio C# Express 2013 with the NuGet package manager enabled
2. .NET 4.5 or later

Implementation Notes

Using Connection Pooling with the C# Client

The connection pool stores open, previously used connections for reuse so your client does not have to negotiate opening a socket for every request. You configure connection pooling for the C# client using the maxStoredConnections parameter in the ScspClient class. For best results, set the value of this parameter to the number of threads multiplied by the number of Swarm cluster nodes.

> Best practice
> For installations with a large number of nodes and a high thread count (approximately 100 or more), limit the value to 5 times the number of threads to avoid reaching the client's operating system limits on open file descriptors.

`ScspClient.Close` is required and does not serve as a no-op.

Notes About the C# SDK

Following are things to keep in mind when writing C# code for the SDK:

- C# Write, Update, Append. A Write, Update, or Append using the C# SDK client that encounters an error response, an ScspWebException might be thrown. This can occur with a 400 response from the cluster, or on any error response (code 400 and greater) when using the SCSP Proxy. This behavior is caused by the way that .NET internally handles a connection closing while writing data to a peer. There is no known workaround.
- Connection timeout. The connection timeout provided to ScspClient is used for two purposes: First as a timeout for connecting and individual read/write API calls, and secondly as a basis for a timeout for the overall HTTP request (this is due to .NET behavior; see HttpWebRequest.Timeout for details). You can also use the size of the object being uploaded as a guide for setting the overall timeout for the entire request. For example, if the connection timeout is set to 300 seconds, a 2GB write will have a full request timeout of approximately 600 seconds. Increasing the connection timeout might help alleviate write failures on large objects due to too many retries of cancelled requests.
- .NET support. .NET 4.5 is required in versions 6.1.1 and later of the SDK. Applications using an older version of .NET must install .NET 4.5 prior to upgrading to version 6.1.1 or later.
- Character encoding. If you pass in a URI path, you must escape a backslash character (\) with %5c.
- Range. You can now use 64-bit Ranges with the C# SDK when using .NET 4.5.

Using the C# Client Sample Code

The C# SDK client ships with runnable sample code that demonstrates how to write client applications for Swarm. The

C# sample code is located in the `sdk-zip-extract-dir/caringo-sdk-version-brand/csharp/ScspCSExamples` directory and consists of the following files:

C++ sample code file name	Description
ScspExample.cs	General SCSP client examples.
ScspRemoteExample.cs	Examples of performing SCSP operations on local and remote clusters using the SCSP Proxy.

Exploring the C# Sample Code

The sdk-extract-dir/caringo-sdk-version-brand/doc/examples directory contains commented C# sample code you can view in a web browser. Double-click index-all.html to open the index page in your default web browser. The commented samples include the following:

- Authenticating named objects
- Performing SCSP operations on unnamed mutable and immutable objects
- Using lifepoints and MD5 integrity seals
- Using READ, including validation, ranges, and Content-MD5
- Performing SCSP operations on objects in local and remote clusters using the SCSP Proxy

C# Administrative Override of an Allow Header

This example is not in the sample code provided with the SDK.

```
// This example shows how to execute an update command
// with administrative override.
String DEFAULTADMINREALM = "CAStor administrator";

ScspUpdate uc = client.CreateUpdateCommand(uuid, inputStream,
testDataLength);
uc.UserAgent = ScspClient.CASTOR_ADMIN_AGENT;
ScspAuthentication auth = new ScspAuthentication("admin",
    "ourpwdofchoicehere", DEFAULTADMINREALM);
headers.Authentication = auth;
uc.Headers = headers;
response = uc.Execute();
```

SDK for Java

- Source Directory Structure
- Java Client Implementation Notes
- Using the Java Client Sample Code

- Java Administrative Override of an Allow Header

Source Directory Structure

The Java source is organized as follows in CAStorSDK-src.zip:

- with
  - `caringo`
    - `client`: Main SCSP Client files, including subfolders
      - `examples`: File(s) showing how to use the Scsp Client to perform various SCSP actions.
      - `locate`: Files used to help SCSP clients locate Swarm instances. The ScspClient class uses either ProxyLocator or StaticLocator to track its configured list of hosts. RoundRobinDnsLocator is a base class for the other locator classes. It implements a Locator based on Round-robin DNS. These examples are unsupported and untested.
      - `request`: Files implementing the Java SCSP communication engine.
- `examples/config` has log4j.properties that shows how to use HttpClient log4j logging

Java Client Implementation Notes

Building the Java SDK

The source zip includes a Maven project file for building the Java SDK. The CastorSDK.jar was built using Maven 3.2.5 and Java 7 (jdk 1.7.0_79). The build depends on the following Maven projects:

- testng 6.8
- Apache httpcomponents.httpclient 4.2.5 and httpcomponents 4.4.4. (Although httpclient 4.5.3 is released, it has not yet been validated with the SDK.)
- Apache log4j 1.2.17
- Maven javax.jmdns 3.4.1

To build the uber jar from scratch, run the following at a command prompt:

```
mvn clean package
```

Java Client Recompile Required

Because of internal changes made to the Java SDK client, you must recompile your Java client code using the classes provided with the SDK when upgrading to version 6.1.4 or later.

Using Connection Pooling with the Java Client

The connection pool stores open, previously used connections for reuse so your client does not have to negotiate opening a socket for every request. You configure connection pooling for the Java client using themaxConnectionPoolSize parameter in the ScspClient class. Caringo recommends you set the value of this parameter to the number of threads multiplied by the number of Swarm cluster nodes.

For more information about this parameter, see the Class ScspClient topic in the Javadoc provided with the SDK, located in the /java/CAStorSDK-doc.zip file in the SDK package.

> Note
>
> For installations with a large number of nodes and a high thread count (approximately 100 or more), you should limit the value to 5 times the number of threads to avoid reaching the client's operating system limits

on open file descriptors.

Remote Cluster Replication and Remote Synchronous Write

In order to support movement of objects between multiple Swarm storage clusters, the SCSP storage API provides a SEND method that can be issued against a source cluster to orchestrate the transfer of a single stream to a destination cluster. SCSP also supports a special type of GET request called a GET/retrieve that triggers a destination cluster to retrieve an object from a source cluster. The Java SDK includes several classes and methods that utilize these SCSP methods to facilitate movement of objects between clusters. The `Replicator` class includes methods for SENDing a single stream, for doing a GET/retrieve on a single stream, and for replicating a single stream using SEND or GET/retrieve via the single-stream methods. It also includes a Remote Synchronous Write (RSW) method to write and then immediately replicate a single stream.

When choosing which methods to use for remote replication, the following guidance applies:

- SEND is simpler and faster but requires that all intervening network components (proxies, gateways, firewalls, routers, etc.) must pass through the SEND method, which is a non-standard extension of HTTP 1.1. Network components that do not recognize the method may return a `501 Not Implemented` response. In such instances, you will need to use GET/retrieve instead.
    - For the lifetime of a `Replicator` instance, the `remoteSynchronousWrite` method attempts to use the SEND method, only falling back to GET/retrieve if SEND is not supported.
- SEND only requires your client to have a path to the local cluster. GET/retrieve may require a network route to both the source and the destination clusters.

Notes About the Java SDK

- Error Returned for Locator Empty IP Address Passing an empty IP Address to the ProxyLocator constructor on a RHEL platform fails with a 'Too many SCSP retries' exception instead of the expected IllegalArgumentException. The same behavior returns the expected exception on a Windows platform.
- Character encoding. If you pass in a URI path, you must escape a backslash character (\) with %5c.
- Maximum Open Connections The maxConnection parameter in the Java implementation of the SDK configures the maximum total open connections, which is the sum of both stored and running connections.
- ResettableFileInputStream FileInputStream() cannot be used; instead, use ResettableFileInputStream, which is located in java\com\caringo\client.
- InputStreams must support reset(); (that is, markSupported() must return true).
- OutputStream Cannot be Null On reads, the value you provide for OutputStream cannot be null. If null is provided, an IllegalArgumentException is thrown.

Using the Java Client Sample Code

The Java SDK client ships with runnable sample code that demonstrates how to write client applications for Swarm. The Java sample code is located in subdirectories of `sdk-zip-extract-dir/caringo-sdk-version-brand/java/com/caringo`:

Subdirectory	Java sample code file name	Description

client/examples	• Environment.java • ScspExample.java • SCSPRemote-Example.java	• Sets the environment to run all SDK code examples. • General client examples. • Client examples that use the SCSP Proxy with local and remote clusters.
realm/examples	RealmExamples.java	Example of creating a domain and bucket.

Exploring the Java Sample Code

The sdk-extract-dir/caringo-sdk-version-brand/doc/examples directory contains commented Java sample code you can view in a web browser. Double-click index- all.html to open the index page in your default web browser. The commented samples include the following:

- Authenticating named objects
- Performing SCSP operations on unnamed mutable and immutable objects
- Using lifepoints and MD5 integrity seals
- Using READ, including validation, ranges, and Content-MD5
- Performing SCSP operations on objects in local and remote clusters using the SCSP Proxy
- Creating domains and buckets

Compiling and Running the Java Code Examples

Before continuing, make sure you completed the tasks discussed in “Using SDK Code Examples” in the SDK Overview.

To get started:

1. If you have not already done so, extract the SDK .zip file into an empty directory.
   This directory is referred to as sdk-extract-dir.
2. If you have not already done so, extract the Java source into the following directory:
   `sdk-extract-dir/java`
3. Open a command prompt window.
4. Set your local environment by either editing the following file or setting local environment variables SCSP_HOST, SCSP_PROXY_HOST, SCSP_PORT, PUBLISHER_HOST, and PUBLISHER_PORT:
   `java-source-code-extract-dir/com/caringo/client/examples/Environment.java`
5. If your cluster administrator changed the default CAStor administrator password, you must edit the example file before compiling and running it. The default password is `ourpwdofchoicehere`. Search for that string in the example files and change it before compiling and running the example.
6. To make setting the classpath easier, set a local environment variable BASE_DIR to the following directory:
   `sdk-extract-dir/java`

Compiling the Java Examples

This section discusses how to compile all of the Java examples. Compile only the examples you intend to run.

To compile the Java examples:

1. Add the uber-jar to your classpath, using either a local environment variable or the `-classpath` *`classpath`* opti on on the `javac` command line:

```
$BASE_DIR
```

2. If you set your environment by editing `Environment.java`, compile it as follows:

```
javac Environment.java
```

3. To compile the SCSP client and remote examples, change to the `sdk-extract-dir/java/com/caringo/client/examples` directory and compile SCSPClientExample.java as follows:

```
javac SCSPClientExample.java
```

4. Compile SCSPRemoteExample.java as follows:

```
javac SCSPRemoteExample.java
```

5. To compile the multi-tenancy examples, change to the `sdk-extract-dir/java/com/caringo/realm/examples` directory and compile RealmExamples.java as follows:

```
javac RealmExamples.java
```

Running the Java Examples

This section discusses how to run the Java examples after you compile them as discussed in the preceding section.

1. To run the SCSP client and remote examples, change to the `sdk-extract-dir/java/com/caringo/client/examples` directory and enter the following command:

```
java com.caringo.client.examples.SCSPClientExample
```

2. Run SCSPRemoteTest.java as follows:

```
java com.caringo.client.examples.SCSPRemoteExample
```

3. To run the multi-tenancy examples, change to the `sdk-extract-dir/java/com/caringo/realm/examples` directory and enter the following command:

```
java com.caringo.realm.examples.RealmExamples
```

Basic Example Troubleshooting

Use the following tips to troubleshoot any errors that you might encounter running the examples:

Some SCSPClientExample examples are expected to fail

For example, the first lifepoint example creates an object with an immediate delete, so subsequent methods on that object fail. An example follows:

```
>>>>> LP Example <<<<<<
Set a terminal lifepoint to delete. Note that the stream will be deleted
as soon as it's written.
LP: [] delete
...[commands omitted]  >>>>>> Read <<<<<<<<
HTTP/1.1 404 Not Found
Date: Fri, 02 Dec 2011 15:49:38 GMT
Content-Length: 92
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
Content-Type: text/html
Server: CAStor Cluster/6.0.0
<html><body><h2>CAStor Error</h2><br>Requested stream was not found
(ENOENT)</body></html>

Retries: 0
ResultCode: ScspRCFailure
```

412 (Precondition Failed)

i412 (Precondition Failed) is expected in some SCSPClientExample ETag examples, such as the following:

```
>>>>>> Read ETAG <<<<<<<<
HTTP/1.1 412 Precondition Failed
Date: Fri, 02 Dec 2011 15:49:39 GMT
Content-Length: 77  Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
Last-Modified: Fri, 02 Dec 2011 15:49:39 GMT
Etag: "75cfb22dd55b800238367a905c927040"
Connection: close
Content-Type: text/html  Server: CAStor Cluster/6.0.0
<html><body><h2>CAStor Error</h2><br>Stream has been
modified</body></html>
  Retries: 0  ResultCode: ScspRCFailure
```

This example fails because the object was changed by the sample code.

401 (Unauthorized)

401 (Unauthorized) is expected in some SCSPClientExample authentication examples, such as the following:

```
>>>>>> Info Domain without Credentials<<<<<< HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="localusers.realm/_administrators",
nonce="2a05cf4e8ff625f5e06bf12aefb0ce86",
opaque="e8b019c6b78eb5137b3deb8aabe88cb0",
stale=false, qop="auth", algorithm=MD5
WWW-Authenticate: Basic realm="localusers.realm/_administrators"
Date: Fri, 02 Dec 2011 15:50:29 GMT
Content-Length: 0  Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
Server: CAStor Cluster/6.0.0
   Retries: 0  ResultCode: ScspRCFailure
```

This failure is expected because the method requires authentication but no credentials were supplied.

Domains

SCSPClientExample examples that depend on domains being set up fail if the domains are not set up properly. Examples follow:

```
Realm 'allusers.realm' doesn't appear to have been set up correctly: Castor-Authorization
header doesn't look right. Returning.
```

The preceding error indicates the allusers.realm was set up with the wrong domain protection setting.

```
Couldn't find realm 'localusers.realm'. Returning.
```

The preceding error indicates the realm does not exist.

Review the information discussed in “Using SDK Code Examples” in the SDK Overview.

SCSP Proxy

In the event the SCSP Proxy is not set up properly or is unavailable, Java exceptions display, including the following:

```
com.caringo.client.ScspExecutionException: Too many SCSP retries.
```

Java Administrative Override of an Allow Header

This example is not included in the code samples provided with the SDK.

```
// This example shows how to execute a copy command with administrative
// override to clean up Allow headers. Note that you may want to get
the
// original headers first to make sure none of the existing ones on the
stream
// are lost.

ScspHeaders headers = new ScspHeaders();
headers.addValue("Allow", "PUT, APPEND"); // we're fixing the Allow
// set the credentials
ScspAuthentication auth = new ScspAuthentication("admin",
"ourpwdofchoicehere", DEFAULTADMINREALM, "");
headers.setAuthentication(auth);
ScspQueryArgs args = new ScspQueryArgs();
// make the request administrative args.setValue("admin", "yes");
response = client.copy(uuid, "", args, headers);
```

SDK for Python

- Installing the Python Client
- Client Startup Behavior
- Using Connection Pooling with the Python Client
- Implementation Notes
- Using the Python Client Sample Code
- Python Administrative Override of an Allow Header

Installing the Python Client

Before you install the Python client, you must install the following prerequisites:

- Python 2.5 or 2.6
  Caringo has tested the Python SDK with the preceding versions of Python. Using other versions might have unpredictable results.
- Python setuptools package for the version of Python you are using

Installing the Python SDK on Python 2.5 or 2.6

The Python client is packaged in an easy to install egg file (python castorsdk-version- pyversion.egg) that contains both source and compiled code. To install the egg on a supported version of Python, enter the following command as a user with root privileges:

```
easy_install castorsdk-version-pyversion.egg
```

The easy install might print some errors related to not finding an index page or not finding a suitable distribution path. These errors can be safely ignored.

Client Startup Behavior

ScspClient.start() is an empty (pass) method. Client startup is implicit in the execution of a request using an ScspClient execution method. This behavior is different from other language implementations of the SDK, which fail when a request is issued after a stop. It is still important that the SDK Python client call ScspClient.stop() at the end of execution so the SDK can clean up any cached open connections.

Using Connection Pooling with the Python Client

The connection pool stores open, previously used connections for reuse so your client does not have to negotiate opening a socket for every request. You configure connection pooling for the Python client using the maxSavedConnectionsparameter in the ScspClient class. Caringo recommends you set the value of this parameter to the number of threads multiplied by the number of Swarm cluster nodes.

> Note
> For installations with a large number of nodes and a high thread count (approximately 100 or more), you should limit the value to 5 times the number of threads to avoid reaching the client's operating system limits on open file descriptors.

Implementation Notes

- Character encoding. If you pass in a URI path, you must escape a backslash character (\) with %5c.
- ScspIOError exception class. In SDK version 6.0 and later, there is an exception class (ScspIOError, derives from IOError) that bypasses the normal retry logic when there is an error reading body data. This class throws an error anytime an object read is attempted but cannot be completed.

Using the Python Client Sample Code

The Python SDK client ships with runnable sample code that demonstrates how to write client applications for Swarm. The Python sample code is located inside the egg file. If you unzip the egg file, the examples are located in the `sdk-zip-extract-dir/caringo-sdk-version-brand/python/castorsdk/examples` directory and consists of the following files:

Python sample code file name	Description
Environment.py[c]	Sets the environment to run all SDK code examples.
ScspExample.py[c]	General client examples.
SCSPRemoteExample.py[c]	Client examples that use the SCSP Proxy with local and remote clusters.
ObjectEnumerator-Example.py[c]	ObjectEnumerator class examples for use with the Content Router. Content Router has been deprecated and these examples will be removed in a future release.
RealmExamples.py[c]	Example of creating a domain and bucket.

Exploring the Python Sample Code

The sdk-extract-dir/SwarmSDK-version/docs/example directory contains commented Python sample code you can view in a web browser. Double-click index-all.html to open the index page in your default web browser. The commented samples include the following:

- Authenticating named objects
- Performing SCSP operations on unnamed mutable and immutable objects
- Using lifepoints and MD5 integrity seals
- Using READ, including validation, ranges, and Content-MD5
- Performing SCSP operations on objects in local and remote clusters using the SCSP Proxy

Creating domains and buckets

Preparing to Run the Python Code Examples

To get started:

1. If you have not already done so, extract the SDK .zip file into an empty directory. This directory is referred to as s dk-extract-dir.
2. If you have not already done so, extract castorsdk-version-pyversion.egg into the following directory: `sdk-extract-dir/python`
3. Open a command prompt window.
4. Set your local environment by either editing the following file or setting local environment variables SCSP_HOST, SCSP_PROXY_HOST, SCSP_PORT, PUBLISHER_HOST, and PUBLISHER_PORT:

```
Python-source-code-extract-dir/python/castorsdk/examples/environmen
t.py[c]
```

5. If your cluster administrator changed the default CAStor administrator password, you must edit the example file before compiling and running it. The default password is ourpwdofchoicehere. Search for that string in the example files and change it before compiling and running the example.

Running the Python Examples

This section discusses how to run the Python examples after you compile them as discussed in the preceding section.

1. Set a local environment variable PYTHONPATH to this directory:
   `sdk-extract-dir/python/castorsdk`
2. To run all examples except the multi-tenancy examples, change to the `sdk-extract-dir/python/castorsdk` directory and enter the following command:

   ```
   python examples/scspExamples.py
   ```

3. Run the remote cluster examples as follows:

   ```
   python examples/scspRemoteExamples.py
   ```

4. Set a local environment variable PYTHONPATH to the `sdk-extract-dir/python` directory.
5. To run the multi-tenancy examples, change to the sdk-extract-dir /python directory and enter the following command:

   ```
   python castorsdk/examples/realmExamples.py
   ```

Basic Example Troubleshooting

Use the following tips to troubleshoot any errors that you might encounter running the examples:

Some scspExamples are expected to fail.

For example, the first lifepoint example creates an object with an immediate delete, so subsequent methods on that object fail. An example follows:

```
>>>>> LP Example <<<<<<
Set a terminal lifepoint to delete. Note that the stream will be deleted
as soon as it's written.
...[commands omitted]
>>>>>> Read <<<<<<
Status Line HTTP/1.1 404 Not Found Status Code 404
Headers
Content-Length: 92
Content-Type: text/html
Date: Sat, 03 Dec 2011 00:00:54 GMT
Server: CAStor Cluster/6.0.0
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE

Response Body HTTP/1.1 404 Not Found
Content-Length: 92 Content-Type: text/html
Date: Sat, 03 Dec 2011 00:00:54 GMT
Server: CAStor Cluster/6.0.0
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE

<html><body><h2>CAStor Error</h2><br>Requested stream was not found
(ENOENT)</body></html>

Result Code 0
Retry Count 0
Root error None
```

412 (Precondition Failed)

412 (Precondition Failed) is expected in some SCSPClientExample ETag examples, such as the following:

```
>>>>>> Read Etag <<<<<<

Status Line HTTP/1.1 412 Precondition Failed Status Code 412
Headers
Last-Modified: Sat, 03 Dec 2011 00:00:56 GMT
Etag: "668897166a53016eb8b3792a0d12a87d"
Content-Length: 77
Content-Type: text/html
Date: Sat, 03 Dec 2011 00:00:56 GMT
Server: CAStor Cluster/6.0.0
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
Connection: close

Response Body HTTP/1.1 412 Precondition Failed
Last-Modified: Sat, 03 Dec 2011 00:00:56 GMT
Etag: "668897166a53016eb8b3792a0d12a87d"
Content-Length: 77
Content-Type: text/html
Date: Sat, 03 Dec 2011 00:00:56 GMT
Server: CAStor Cluster/6.0.0
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE
Connection: close
<html><body><h2>CAStor Error</h2><br>Stream has been
modified</body></html>

Result Code 0
Retry Count 0
Root error None
```

This example fails because the object was changed by the sample code.

401 (Unauthorized)

401 (Unauthorized) is expected in some SCSPClientExample authentication examples, such as the following:

```
>>>>>> Info named object 'protectedobject' without credentials<<<<<<
Status Line HTTP/1.1 401 Unauthorized
Status Code 401
Headers
Content-Length: 0
WWW-Authenticate: Digest realm="allusers.realm/2814339560",
nonce="2a05cf4e8ff625f5e06bf12aefb0ce86",
opaque="97024db82e5e3f455e845cb89ee89e08", stale=false, qop="auth",
algorithm=MD5
WWW-Authenticate: Basic realm="allusers.realm/2814339560"
Date: Sat, 03 Dec 2011 00:01:51 GMT
Server: CAStor Cluster/6.0.0
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE

Response Body HTTP/1.1 401 Unauthorized
Content-Length: 0
WWW-Authenticate: Digest realm="allusers.realm/2814339560",
nonce="2a05cf4e8ff625f5e06bf12aefb0ce86",
opaque="97024db82e5e3f455e845cb89ee89e08", stale=false, qop="auth",
algorithm=MD5
WWW-Authenticate: Basic realm="allusers.realm/2814339560"
Date: Sat, 03 Dec 2011 00:01:51 GMT
Server: CAStor Cluster/6.0.0
Allow: HEAD, GET, PUT, POST, COPY, APPEND, DELETE

Result Code 0
Retry Count 0
Root error None
```

This failure is expected because the method requires authentication but no credentials were supplied.

Domains

scspExamples that depend on domains being set up fail if the domains are not set up properly. Examples follow:

```
Realm 'allusers.realm' doesn't appear to have been set up correctly:
Castor-Authorization header doesn't look right. Returning.
```

The preceding error indicates the allusers.realm was set up with the wrong domain protection setting.

```
Couldn't find realm 'localusers.realm'. Returning.
```

The preceding error indicates the realm does not exist.
Review the information discussed in SDK Overview and try running the examples again.

SCSP Proxy

In the event the SCSP Proxy or Content Router are not set up properly or are unavailable, Python exceptions display, including the following:

```
Can't write to proxy host. It looks like I can't talk to your proxy or its remote server.
Please check your configuration and retry.
```

Python Administrative Override of an Allow Header

This example is not included in the code samples provided with the SDK.

```
# This example shows how to execute a copy command with administrative
# override to clean up Allow headers. Note that you may want to get the
# original headers first to make sure none of the existing ones on the
stream
# are lost.

DEFAULTADMINREALM = 'CAStor administrator'

#set admin credentials
auth = ScspAuthentication()
auth.realm = DEFAULTADMINREALM #realm doesn't matter
auth.cnonce = 'abcdef' # actual value doesn't matter
auth.user = 'admin'
auth.password = 'ourpwdofchoicehere'
headers = ScspHeaders()
headers.addValue('Allow', 'PUT')
headers.authentication = auth
args = ScspQueryArgs()
#make this request administrative
client.copy(uuid, args, headers)
```